Structured Dictionary Learning and its applications in Neural Recording

by

Yuanming Suo

A dissertation submitted to The Johns Hopkins University in conformity with the requirements for the degree of Doctor of Philosophy.

Baltimore, Maryland

July, 2015

Abstract

Widely utilized in the field of neuroscience, implantable neural recording devices could capture neuron activities with an acquisition rate on the order of megabytes per second. In order to efficiently transmit neural signals through wireless channels, these devices require compression methods that reduce power consumption. Although recent Compressed Sensing (CS) approaches have successfully demonstrated their power, their full potential is yet to be explored, particularly towards exploring a more efficient representation of the neural signals. As a promising solution, sparse representation not only provides better signal compression for bandwidth/storage efficiency, but also leads to faster processing algorithms as well as more effective signal separation for classification purpose. However, current sparsity-based approaches for neural recording are limited due to several critical drawbacks: *(i)* the lack of an efficient data-driven representation to fully capture the characteristics of specific neural signal; *(ii)* most existing methods do not fully explore the prior knowledge of neural signals (e.g., labels), while such information is often known; and *(iii)* discriminative information is not encoded into the representation to promote classification.

ABSTRACT

Using neural recording as a case study, this dissertation presents new theoretical understandings and mathematical frameworks on structured dictionary learning with applications in compression and classification. We begin by showing that using a data dictionary can significantly compress the neural data thanks to its self-similarity. Under a single task setup, we provide theoretical proofs to show the benefits of using structured sparsity in dictionary learning. We provide various models for the representation of a single measurement and multiple measurements, where signals exhibit both with-in class similarity as well as with-in class difference. Under the assumption that the label information of the neural signal is known, the proposed models minimize the data fidelity terms together with structured sparsity terms to yield a more discriminative representation. We demonstrate that this is particularly essential in neural recording since it can improve the compression ratio, classification accuracy and help deal with non-ideal scenarios such as co-occurrences of neuron firings. Fast and efficient algorithms based on Bayesian inference and alternative direction method are proposed. Extensive experiments are conducted on both neural recording applications as well as other classification task, such as face recognition.

Primary Reader: Professor Trac D. Tran

Secondary Reader: Professor Ralph Etienne-Cummings

Acknowledgments

First and foremost, I would like to thank my advisor Prof. Trac D. Tran who has always been a wise advisor, a caring mentor, and a dearest friend throughout my entire PhD program. He led me to explore interesting mathematical concepts, gave me great freedom in choosing research topics and shared with me so much of his life wisdom to help me live up to my potential. This thesis would never have been done without his guidance, support, and encouragement.

I thank my dissertation committee members Prof. Ralph Etienne-Cummings and Prof. Mark A. Foster for their time, interest, and constructive suggestions to this dissertation. I also thank Prof. Sang Peter Chin from the Boston University for his insightful research advice, and Prof. Vishal Monga, Dr. Umamahesh Srinivas and Hojjat Seyed Mousavi at the Penn State University for fruitful discussions on structured dictionary learning methods. I would like to show my sincere and genuine gratitude to Prof. Larry D. Carey and Prof. Reza Adhami, my advisors during my master's program at the University of Alabama in Huntsville. They directed me to the right path, and helped me develop good habits and skills.

ACKNOWLEDGMENTS

It has been a great pleasure to be a member of the DSP lab. I would like to thank all of the following lab members for their help and numerous exciting discussions: Dr. Dzung T. Nguyen, Dr. Nam H. Nguyen, Dr. Yi Chen, Dr. Minh D. Dao, Dung Tran, Qing Qu, Xiaoxia Sun, Sonia Joy, Tao Xiong, Xiang Xiang, Luoluo Liu and Akshay Rangamani. I am also grateful to have the opportunities to work with passionate and intelligent collaborators at the Johns Hopkins University: Dr. Emi Z. Murano, Dr. Jonghye Woo, Prof. Maureen Stone, Prof. Jerry L. Prince, and Dr. Garrick Orchard. I want to give special thanks to Jie Zhang, with whom I started developing cool ideas around neural recording and compressive video camera. I am also grateful to my friends who share my joy and help me tremendously through the hard time. They are the greatest treasure I have ever had.

I gratefully acknowledge the funding sources that made my Ph.D. work possible: the Electrical and Computer Engineering department at JHU, the National Science Foundation, and the Army Research Office.

I owe my deepest gratitude to my parents for their unconditional love and support in all my pursuits. They give me the ultimate freedom to choose the path I want and always believe that one day I can fulfill my dream. Finally, I want to thank Qiuyuan Liu, my fiancee, who has always been there cheering me up and standing by me through the good times and bad. My greatest achievement during the Ph.D. years is learning the meaning of true love together with you.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Implantable neural recording devices, such as Multi-electrode arrays (MEA), have been widely used by neuroscientists to monitor the neural activities within designated brain areas. With bandwidths up to 10 kHz, these neural signals are often sampled at a frequency above 20 kHz as mandated by the Nyquist sampling theorem. Since the signal usually takes a resolution above 10 bits, the acquisition rate of an MEA with hundreds of electrodes (i.e., a Utah array) is on the order of megabytes per second. This high acquisition rate poses a significant challenge for transmitting the signal off-chip, especially using wireless communications, where the induced power consumption is in the mW range for traditional approaches.[2] Thus, most MEAs are only utilized in a highly restricted experimental setup, in which either the number of electrodes is limited or wired communication is employed.

To tackle the challenges faced by the neural recording devices, we proposed new

mathematical models and algorithms to enforce various structured sparsity constraints in a dictionary learning framework. To illustrate its benefits for neural signal compression and classification, we organize the dissertation as follows:

Chapter 1 presents an overview of our contributions.

Chapter 2 introduces mathematical concepts of sparse representation, compressed sensing and dictionary learning. We also specify the notations used throughout the dissertation.

Chapter 3 presents our first attempt to use data-driven dictionary in conjunction with compressed sensing for implantable neural recordings. Our approach includes designing of the sparsifying dictionary, a two-layer sensing strategy as well as a sparse recovery method using Spike and Slab prior.

Built upon our on-chip compressed sensing implementation, we propose an energy efficient multi-mode CS framework that focuses on improving the off-chip components of neural recording device, including *(i)* a two-stage sensing strategy, *(ii)* a sparsifying dictionary directly using data, *(iii)* enhanced compression performance from Full Signal CS mode to Spike CS + Restoration mode and; *(iv)* extension of our framework to the Tetrode CS recovery using joint sparsity. This new framework achieves energy efficiency, implementation simplicity and system flexibility simultaneously. Extensive experiments are performed on simulation and real datasets. For our Spike CS + Restoration mode, we achieve a compression ratio of 6% with a reconstruction *SNDR* $>$ 10dB and a classification accuracy $>$ 95% for synthetic datasets. For real datasets,

we get a 10% compression ratio with $\sim$ 10dB for Spike CS + Restoration mode.

Chapter 4 introduces our structured dictionary learning framework, including both Hierarchical Dictionary Learning (HiDL) for a single task setup and Group Structured Dirty Dictionary Learning (GDDL) for an multi-task scenario. We further demonstrate the superiority of using structured sparsity for classification through theoretical analysis of its performance.

In many areas of science and engineering beyond neural recording, researchers are dealing with signals that are often inherently sparse with respect to a certain dictionary (also called basis or transform). The seminal paper by neuroscientists Olshausen and Field[3] points out that the receptive fields in human being's visual cortex utilize sparse coding to extract meaningful information from images. To better capture the data characteristics, various dictionary learning methods have been proposed for both reconstruction and classification tasks. For classification particularly, most approaches proposed so far have focused on designing explicit constraints on the sparse code to improve classification accuracy while simply adopting ℓ_0-norm or ℓ_1-norm for sparsity regularization. Motivated by the success of structured sparsity in the area of Compressed Sensing, we propose Hierarchical Dictionary Learning (HiDL) and generalize it to Group Structured Dirty Dictionary Learning (GDDL). The latter incorporates the structure information on both group and task levels in the learning process. Its benefits are two-fold: *(i)* the label consistency between dictionary atoms and training data is implicitly enforced, and *(ii)* the classification performance is more

robust than other techniques in the case of a small dictionary size or limited training data. Using the subspace model, we derive the conditions for HiDL to guarantee the performance and show theoretically that using structured sparsity is superior to ℓ_0-norm or ℓ_1-norm regularized dictionary learning for classification.

Chapter 5 includes extensive experiment results of using HiDL and GDDL on synthetic simulation, neural recording, face recognition and object classification datasets.

After developing the theoretical frameworks and algorithms of HiDL and GDDL, we apply these approaches to synthetic simulations, neural recording, and other real-world applications, such as face recognition and object classification, to demonstrate the validity of the proposed DL framework. Through extensive experiment results on neural recording, we show that using HiDL can improve the recovery performance while significantly boosting classification performance when the compression ratio is very low (e.g., 5%). Moreover, to suit for the situation that dictionary needs to be learned directly from the compressed measurements rather than the original signal, we develop an extension of HiDL – HiDL-CS. HiDL-CS could learn a dictionary from the randomly compressed measurements as long as the compression ratio is not too low. Meanwhile, the structured information incorporated in the sparse coefficients allows HiDL-CS to yield comparable classification accuracy to HiDL even when its recovery performance might not be as good. This provides an alternative approach for learning the dictionary without interrupting compression. Finally, we demonstrate that GDDL could be used to address the non-ideal situation of the co-occurrences

of the neuron firings. In this case, a signal frame may contain two or more spikes and they might be super-positioned on top of each other. GDDL could effectively separate the consistent neuron firings (the with-in class similarity) from the sparsely firing of arbitrary neurons (the with-in class difference). Thus, we could use the clean dictionary of the desired neurons to accurately classify the signal with high recovery performance. Besides the case of neural recording, we also demonstrate the classification performance of the proposed HiDL and GDDL in synthetic and real datasets.

Chapter 6 concludes the dissertation with a discussion of ongoing work and future directions.

Chapter 2

Background

According to The Economist, the global digital information is projected to be more than 34.6 Zettabytes in 2020, three times more than the projected storage capacity. More than 90% of this Big Data is unstructured with a very large portion contributed by the wearable sensors. By 2020, it is projected that every human being will contribute 1,000 Gigabytes of the sensor data individually. These sensors include but not limited to GPS, accelerometer, gyroscope, microphone, camera, all kinds of biomedical signal sensors, etc. Beyond the domain of monitoring human activity, sensor data processing has been an active research topic within the context of numerous practical applications, such as medical image analysis, remote sensing, and military target/threat detection.

One powerful tool to tackle these critical Big Data problems is signal-processing techniques based on sparse representation.[4] A sparse representation not only provides

better signal compression for bandwidth/storage efficiency, but also leads to faster processing algorithms as well as more effective signal separation for detection, classification and recognition purposes because it focuses on the most intrinsic property of the data. Sparse signal representation allows us to capture the hidden simplified structure present in the data jungle, and thus minimizes the harmful effects of noise in practical settings.

2.1 Sparse Representation

Sparse representation (SR) has been rigorously studied over the past few years as a powerful signal processing paradigm. According to the SR theory, a signal $\mathbf{x} \in \mathbb{R}^N$ can be represented using an s-sparse coefficient vector $\mathbf{a}^\star \in \mathbb{R}^K$ with respect to a dictionary matrix $\mathbf{D}$ of size $N \times K$, where s-sparse means that the number of non-zero coefficients of $\mathbf{a}^\star$ is no more than s. To reconstruct $\mathbf{a}^\star$, the following ℓ_1-minimization problem (or Lasso) is proposed:[5–7]

$$\text{Noiseless:} \quad \min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{s.t.} \quad \mathbf{x} = \mathbf{D}\mathbf{a}. \tag{2.1.1}$$

To handle the noisy case with imperfect representations contaminated by bounded Gaussian noise, the formulation becomes:

$$\text{Noisy:} \quad \min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{s.t.} \quad \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_2 \leq \sigma, \tag{2.1.2}$$

where σ is the standard deviation of the zero mean Gaussian noise.

Structured sparse representation (SSR) could yield better results than using ℓ_1-norm as in (2.1.1) and (2.1.2) for applications with additional prior information. With known hierarchical structure, Group Lasso[8,9] penalizes the group level sparsity. However, Group Lasso tends to produces results that are dense inside each group. Thus, Hierarchical Lasso (HiLasso) is proposed to regualize both the group sparsity and in-group sparsity.[10] And its multi-task version Collaborative HiLasso (C-HiLasso) takes into account of both the group structure in each task and block struture across multiple tasks. In the case of multiple measurements $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_T]$ capturing similar events, the correlation between observations in the sparse representation can be reinforced by joint sparsity, which imposes the same sparsity patterns on the sparse coefficients:[11]

$$\text{Joint Sparsity:} \quad \min_{\mathbf{A}} ||\mathbf{A}||_{12} \quad \text{s.t.} \quad \mathbf{X} = \mathbf{DA}, \tag{2.1.3}$$

where $\mathbf{A} = [\mathbf{a}_1, ..., \mathbf{a}_T]$ and the norm $||\mathbf{A}||_{12}$ is defined as the sum of the ℓ_2-norm of different rows of $\mathbf{A}$.

2.2 Compressed Sensing

Compressed sensing (CS) introduces a theoretical framework regarding the exact recovery of a signal $\mathbf{x}$ from its compressed measurement vector $\mathbf{y} \in \mathbb{R}^M$ under the assumption that $\mathbf{x}$ is s-sparse (where $s < M << N$). Given that a sensing matrix $\mathbf{S}$

satisfying the Restricted Isometry Property (RIP) and $M \sim s\log(\frac{N}{s})$,[5,6] the s-sparse vector $\mathbf{x}$ can be recovered with high probability by solving the following ℓ_1-norm minimization problem:

$$\min_{\mathbf{x}} ||\mathbf{x}||_1 \quad s.t. \quad ||\mathbf{y} - \mathbf{S}\mathbf{x}||_2 \leq \sigma, \tag{2.2.1}$$

where $\mathbf{S} \in \mathbb{R}^{M \times N}$ is the sensing matrix. Random Gaussian and random Bernoulli matrices have been shown to satisfy RIP with very small M regardless of the choice of the sparsifying dictionary. Variants of Bernoulli, such as punctured Bernoulli circulant matrix, are proposed to further reduce the hardware complexity.[12] Most often, the signal is not sparse in time domain (i.e., image and video) but with respect to some basis $\mathbf{D}$ as pointed out in previous section. Then the optimization problem becomes:

$$\min_{\mathbf{a}} ||\mathbf{a}||_1 \quad s.t. \quad ||\mathbf{y} - \mathbf{S}\mathbf{D}\mathbf{a}||_2 \leq \sigma, \tag{2.2.2}$$

where we replace the original signal $\mathbf{x}$ with its sparse representation $\mathbf{D}\mathbf{a}$ and σ controls the quality of approximation. After recovering the s-sparse signal $\mathbf{a} \in \mathbb{R}^K$, the estimate of signal $\mathbf{x}$ can be recovered by:

$$\hat{\mathbf{x}} = \mathbf{D}\mathbf{a}. \tag{2.2.3}$$

The recovery performance of CS is strongly related to the design of sensing matrix

and sparsifying dictionary. It has been shown in[13] that *mutual-coherence* can be used to compare different sensing matrices using the same dictionary. The *mutual-coherence* $\mu(\mathbf{S}, \mathbf{D})$ is defined as the largest absolute value of the normalized inner products between different columns of the design matrix $\mathbf{E} = \mathbf{SD}$, which can be formally written as:

$$\mu(\mathbf{S}, \mathbf{D}) = \max_{1 \leq i,j \leq K, i \neq j} \frac{|\mathbf{e}_i^T \mathbf{e}_j|}{||\mathbf{e}_i|| \cdot ||\mathbf{e}_j||}, \tag{2.2.4}$$

where $\mathbf{e}_i$ is the i-th column of $\mathbf{E}$. In general for a given dictionary, the matrix $\mathbf{S}$ that attains a smaller *mutual-coherence* can achieve the same reconstruction performance with a smaller measurement number M. However, minimizing *mutual-coherence* involves calculating all pair-wise inner products and is computationally expensive. Thus, Elad[14] proposed to optimize the sensing matrix $\mathbf{S}$ by iteratively reducing the t-averaged *mutual-coherence*, which minimizes the correlation larger than certain threshold. In,[15] Sapiro *et al.* proposed to optimize $\mathbf{S}$ by reducing the average *mutual-coherence* and is shown to achieve better performance than.[14] Both of these approaches perform better than random Bernoulli matrix. However, using them as the alternative on-chip sensing matrix will increase the complexity for circuit implementations because of their fractional values.

2.3 Dictionary Learning

The assumption for using SR is that the signal is sparse with respect to a certain deterministic dictionary. Traditionally, dictionaries are designed to incorporate desired properties in time/space or frequency domains, or a mixture of both such as wavelets. Recently, a different methodology is explored to learn the dictionary directly from the data to better capture its characteristics. Initially, the dictionary learning (DL) method is designed primarily for reconstruction:

$$\text{DL for Reconstruction:} \quad \min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1||\mathbf{a}_i||_q). \tag{2.3.1}$$

Given the training data $\mathbf{x}_i \in \mathbb{R}^N$ $(i = 1, ..., I)$, a dictionary $\mathbf{D} \in \mathbb{R}^{N \times K}$ and the corresponding sparse coefficients $\mathbf{a}_i$ are learned. The regularizer ℓ_q-norm is used to promote sparsity, which could be ℓ_0-norm,[16,17] ℓ_1-norm,[18–20] ℓ_2-norm with locality constraint,[21] structured sparsity,[22,23] or sparsity promoting prior.[24]

DL could also be interpreted as a mapping between the low dimensional signal $\mathbf{x}$ and its high dimensional sparse feature $\mathbf{a}$. Thus, it has also been employed for discriminative tasks, such as classification. To design the dictionary and sparse code with discriminating properties, extra constraints $f_{\mathbf{A}}(\cdot)$ and $f_{\mathbf{D}}(\cdot)$ are enforced, leading

to:

$$\text{DL for Discrimination:} \quad \min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1||\mathbf{a}_i||_q) + \lambda_2 f_{\mathbf{A}}(\mathbf{A}) + \lambda_3 f_{\mathbf{D}}(\mathbf{D}). \tag{2.3.2}$$

The candidates for function $f_{\mathbf{A}}(\cdot)$ could be logistic functions,[25,26] linear classifier,[27] label consistency,[28] or Fisher discrimination criterion.[29] An example of $f_{\mathbf{D}}(\cdot)$ is to force the dictionaries of different classes to be as incoherent as possible.[30] The classification decision could be obtained from residue with respect to the sub-dictionary of each class,[30] a linear classifier[27,28] or a logistic function.[25,26]

2.4 Notation

In this section, we introduce notations that will be used throughout the article. We use bold lower-case letters such as $\mathbf{x}$ to represent vectors, bold upper-case letters such as $\mathbf{D}$ to represent matrices, and bold lower-case letter with subscript such as $\mathbf{d}_j$ to represent a column of a matrix. The dimensions of vectors and matrices are often clear from the context.

For any vector $\mathbf{a}$, we use $||\mathbf{a}||_q$ to denote its ℓ_q-norm ($0 \leq q \leq \infty$).

For any matrix $\mathbf{A}$, we use $||\mathbf{A}||_{1,1}$, $||\mathbf{A}||_{1,2}$ and $||\mathbf{A}||_{1,\infty}$ to denote its $\ell_{1,1}$-norm, $\ell_{1,2}$-norm[10] and $\ell_{1,\infty}$-norm,[31] respectively. We also use Prox$(\cdot)$ to denote proximal operators, which will be explained in details in Chapter 4. A group g is a subset of indices in $\{1, ..., K\}$. A group structure $\mathcal{G}$ denotes a pre-defined set of non-overlapping

groups. We use $\rho(\cdot)$, $tr(\cdot)$, $rank(\cdot)$, $dim(\cdot)$, and $svd(\cdot)$ to denote spectral norm, trace, rank of the matrix, dimension of the subspace, and singular value decomposition, respectively.

Chapter 3

Energy-Efficient Multi-Mode Compressed Sensing System for Implantable Neural Recordings

Implantable neural recording devices, such as Multi-electrode arrays (MEA), have been widely used by neuroscientists to monitor the neural activities within a designated brain area. With bandwidths up to 10 kHz, these neural signals are often sampled at a frequency above 20 kHz as mandated by the Nyquist sampling theorem. Since the signal usually takes a resolution above 10 bits, the acquisition rate of a MEA with hundreds of electrodes (i.e., a Utah array) is on the order of megabytes per second. This high acquisition rate poses a significant challenge for transmitting the signal off-chip, especially using wireless communications, where the induced power

consumption is in the mW range for traditional approaches.[2] Thus, most MEAs are only utilized in a highly restricted experimental setup, in which either the number of electrodes is limited or wired communication is employed.

3.1 Prior Works and our Contribution

To release the full potential of MEAs, a straightforward strategy is to reduce the high acquisition rate or to compress the data on-chip before transmission. Currently, there are three categories of lossy compression approaches utilized for MEAs: event-based approaches, transformation-based approaches, and Compressed Sensing approaches. Among them, event-based approaches have the simplest implementation. An example of event based approaches is spike detection.[32–36] The spikes are first detected in the neural signal using threshold crossing, then only the small segments containing the spikes are transmitted. This mechanism could be implemented using only a few circuit components, resulting in a small layout area and low power consumption. No off-chip processing is carried out to further process the signal. Therefore, its disadvantage is obvious since the information contained in the segments without spikes, which would be useful for signal analysis, can never be recovered.

If the whole signal is required, a transformation-based approach is usually chosen. A well-known candidate for this category is the on-chip wavelet transform, which yields high compression ratio and good reconstruction quality.[37,38] This approach

takes advantage of the fact that the neural signal can be sparsely represented or approximated with respect to a wavelet dictionary. Therefore, only a small fraction of its significant wavelet coefficients, instead of the neural signal, are transmitted. The off-chip algorithm then takes the inverse wavelet transform to approximate the original signal. However, an ASIC implementation of the required on-chip wavelet transform demands large dedicated DSPs and memory operating above the Nyquist rate of the spikes ($>$ 20 kHz). Moreover, its power consumption is not tunable because the entire wavelet transform always needs to be carried out, hence reducing the flexibility of the system.

Recently, the field of Compressed Sensing (CS) has shown potential in achieving compression and reconstruction performance comparable to the transformation-based approach but with a much simpler circuitry.[2,39–43] The CS approach is based on the same sparsity assumption as the transformation-based approach, but it does not sparsify the signal on-chip, and therefore, avoids the needs of dedicated DSPs. Instead, CS approach acquires a set of the random measurements of the original signal and leaves most of the computational burden to off-chip processing. It is theoretically proven that CS-based systems can perfectly reconstruct a signal using only a small fraction of its noiseless random measurements. Even in the case of small Gaussian noise, the recovered signal is guaranteed to be within a bounded neighborhood of the original signal. The recovery quality and power consumption of CS-based systems are closely related to its compression ratio, thus users could adjust the amount of

measurements to be collected to meet requirements of different applications. Nevertheless, two challenges still remain. First, there is a trade-off between the complexity of the sensing circuit and its compression capability. Second, a careful design of the sparsifying dictionary is needed to guarantee the compression performance.[44]

In summary, we identify three key factors to consider when designing an efficient compression approach for implantable neural recordings:

(i) Energy Efficiency: A system should have a high compression ratio with good reconstruction performance. This could significantly reduce the power consumption. Both transformation-based and CS approaches share this feature.

(ii) Implementation Simplicity: An ideal system should have simple realization of on-chip compression module and efficient off-chip reconstruction algorithms. However, trade-off always exists between complexity of the system and its performance. Both event-based and CS approaches may have simple on-chip implementations while CS approaches require complicated off-chip algorithms to gain better recovery performance (i.e., dictionary learning algorithm in[43]).

(iii) System Flexibility: All aforementioned approaches either provide an estimate of the full signal or the spike segments. It would be ideal if a single system can have multiple working modes providing both full signal and spikes that could be used for different experiment configurations.

Our previous design[43] focused on a simple on-chip implementation of random Bernoulli matrix for the sensing matrix. For off-chip reconstruction, we adopted dic-

tionary learning to train the sparsifying dictionary and then relied on greedy methods to reconstruct the signal (as in Fig. 3.1(a)). Driven by the aforementioned key factors, we propose an energy-efficient multi-mode extension of our CS system. With the same on-chip sensing matrix implementation as in,[43] we focus on the following components of the off-chip design:

(i) Two-stage sensing approach: To address the trade-off between complexity of the sensing matrix and its compression performance, we propose a two-stage CS approach for implantable neural recordings as illustrated in Fig. 3.1(b). Besides the on-chip Bernoulli sensing matrix $\mathbf{S}$, we add a second stage of off-chip sensing using Puffer Transformation $\mathbf{P}$ to further boost the compression performance, and ultimately, the power efficiency.

(ii) Data dictionary: Different from all previously mentioned techniques, we propose to use data directly as the sparsifying dictionary $\mathbf{D}$ in Fig. 3.1(b-d), which is inspired by.[45] As shown later, the data dictionary provides comparable performance to the signal dependent dictionary, but without extra computation of dictionary learning. Moreover, both the off-chip sensing matrix $\mathbf{P}$ and the dictionary $\mathbf{D}$ can be updated incrementally with low computational complexity. This helps us improve the implementation simplicity and enables an efficient neural signal processing system.

(iii) Three working modes: Different from our previous work that compressed the full signal, our new design allows the users to switch between three working modes: *(1)* the full signal going through on-chip CS and off-chip reconstruction as in

Fig. 3.1(b); *(2)* the spike detection followed with off-chip restoration to the full signal as in Fig. 3.1(c); and finally *(3)* the spike detection with on-chip CS and off-chip CS recovery plus restoration to the full signal as in Fig. 3.1(d). As a combination of the first two modes, the last Spike CS + Restoration mode is unique for two reasons, *(1)* it achieves enhanced compression performance because only spike segment rather than full signal is compressed and, *(2)* given CS measurements of only spike segments, the whole signal is recovered to provide information on the non-spike segment as well. The three working modes also give users an all-in-one system with the flexibility to choose the desired mode they need.

(iv) Tetrode CS recovery: All components mentioned above are geared towards a single electrode CS system. Here, we also provide a sparse recovery algorithm using Spike and Slab priors and joint sparsity for the simultaneous CS recovery of the multi-electrode neural signal recording (i.e., Tetrode). We have shown that our system could be easily extended to the Tetrode application with an improved compression performance comparing to the case of performing CS recovery on each electrode independently.

3.2 Our CS framework

3.2.1 Our Sparsifying Dictionary

Physiological recordings suggest that shapes of the spikes are quite reproducible for each neuron over time (as shown in left column of Fig. 3.9). Based on this observation, neuroscientists are able to distinguish multi-neuron activities using spike sorting techniques.[46,47] Therefore, the inherent dimension of the neural signal is much smaller than its ambient dimension. Inspired by this observation, we adopt the concept of *self-expressiveness* from.[45,48] This property assumes that each data point can be sparsely represented as a linear combination of other points in the same subspace, which can be formally written as:

$$\mathbf{d}_i = \mathbf{D}\mathbf{a}_i \quad s.t. \quad ||\mathbf{a}_i||_0 \leq s,\ a_{i,i} = 0,\ \forall\ i, \tag{3.2.1}$$

where $\mathbf{d}_i$ is the i-th column of $\mathbf{D}$, $\mathbf{a}_i$ is the corresponding sparse coefficients, and the i-th coefficient of $\mathbf{a}_i$ is zero so $\mathbf{d}_i$ will not be used to represent itself. For our framework, we take advantage of the similarity of the spikes, and use pre-acquired full signal data as the dictionary $\mathbf{D}$ to represent the other newly acquired full signal in the same subspace. Its benefit is obvious because the intense computation needed by dictionary learning can be waived. Instead, the dictionary $\mathbf{D}$ is built by either periodic acquisition at Nyquist rate or the recovered spikes. In our case, we choose

the first one because it takes very short time to acquire and update the dictionary. In the case of sparsely firing neurons (no prior existence in the dictionary), we trigger the full Nyquist acquisition to update dictionary when the reconstruction performance is not good enough.

3.2.2 Our Sensing Matrix

Different from other CS-based approaches, we propose a two-stage sensing scheme, which includes an on-chip sensing stage with $\mathbf{S}$ and an off-chip sensing stage with $\mathbf{P}$. The optimization problem now becomes:

$$\underset{\mathbf{a}}{\operatorname{argmin}} ||\mathbf{a}||_1 \quad s.t. \quad ||\mathbf{z} - \mathbf{PSDa}||_2 \leq \sigma, \tag{3.2.2}$$

where $\mathbf{P} \in \mathbb{R}^{M \times M}$ is the off-chip sensing matrix and $\mathbf{z} = \mathbf{Py} \in \mathbb{R}^M$ is the measurement after second sensing stage.

3.2.2.1 On-chip Sensing

For on-chip sensing matrix $\mathbf{S}$, we choose a digital implementation of Random Bernoulli Matrix containing values of either 1 or -1 at every entry. Sensing using Random Bernoulli matrix can be implemented using several area and power efficient digital accumulators operating at signal Nyquist rate.[2,43] This implementation is more power and area efficient than the implementation of the Random Gaussian

matrices or Optimized matrices,[14,15] whose implementations require either multi-bit digital multipliers or implementation of multiple DACs and analog integrators.[49]

Fig. 3.2 shows our on-chip sensing implementation in the TSMC 180nm process. The CS circuit for each electrode contains 26 accumulator shift-registers (ASR). The accumulations are clocked at signal Nyquist rate of 20 KHz (C20K). The Matrix block, shared across all the channels, contains 26 registers to hold one row of a random Bernoulli matrix. Their values are updated at every Nyquist period. The ASRs and the matrix block implement matrix multiplication between a signal vector of length N (i.e., $N = 128$) and a Bernoulli matrix having a dimension of N by M (i.e., $M \leq 26$). Depending on the value of a particular matrix entry (either 1 or 0), the corresponding ASR either adds or subtracts the current digitized signal from the accumulated value. To avoid the need of extra registers for buffering the data for transmission, a 4 MHz (C4M) clock is used to shift the data from the ASR to the output pin near the end of accumulation cycle. Each ASR can be disabled by applying clock gating to control the compression ratio. The CS circuit also contains a spike detection block, implemented using a 10-bit full adder, whose output can either be transmitted off-chip or through CS circuit.[43] The On-Chip Sensing blocks function with VDD of 0.53V without performance degradation. The CS block uses 0.11 mm^2 area and 0.83 uW (digital) power per electrode when compressing the signal at a compression ratio of 10%. Since this paper focuses on the off-chip components of our CS framework (i.e., dictionary, off-chip sensing and recovery), more details of

the design and specifications of our chip will be presented in a separate paper.

3.2.2.2 Off-chip Sensing

For the off-chip sensing stage, we adopt the concept of Puffer Transformation from the field of Statistics.[50] Given a design matrix $\mathbf{E} = \mathbf{SD}$, the corresponding Puffer Transformation $\mathbf{P}$ inflates its smallest non-zero singular values, therefore improves the irrepresentable condition, which is related to *mutual-coherence*. Intuitively, the Puffer Transformation maintains the dimension of the on-chip measurement, but adjusts the radius of the ℓ_2-norm ball (data fidelity term) to become a sphere so that the ℓ_1-norm regularized problem is more likely to find the correct solution. Interested readers can refer to[50,51] for more theoretical analysis. If we define the singular value decomposition (SVD) of $\mathbf{SD} = \mathbf{U\Sigma V}^\top$, the corresponding off-chip sensing matrix $\mathbf{P}$ will then be:

$$\mathbf{P} = \mathbf{U\Sigma}^{-1}\mathbf{U}^\top. \tag{3.2.3}$$

Note that if the i-th singular value $\mathbf{\Sigma}_{i,i}$ is zero, then we define $\mathbf{\Sigma}_{i,i}^{-1}$ to be zero as well.

To understand the effect of the second stage off-chip sensing $\mathbf{P}$ on the average *mutual-coherence*, we generate a simulated dataset with $\mathbf{D}$ as random Gaussian matrix of size 100×1000 and the sensing matrix $\mathbf{S}$ of size 20×100. We compare the proposed two-stage CS approach (on-chip Bernoulli and off-chip $\mathbf{P}$) with random Bernoulli, random Gaussian and Sapiro's optimized sensing matrix approach. The

distribution of the normalized pairwise correlation between columns of $\mathbf{E}$ is shown in Fig. 3.3. We use the same Bernoulli sensing matrix for our two-stage sensing approach and the approach with only on-chip Bernoulli sensing to show the effect of having an additional off-chip sensing step. For our approach, the correlation is calculated for the columns of matrix product $\mathbf{PSD}$ rather than $\mathbf{SD}$ for other approaches. We can see that the distributions of both the proposed approach and Sapiro's approach skew towards zero, therefore can achieve a smaller average *mutual-coherence* than that of random Gaussian and random Bernoulli. Thus, they can further improve the CS reconstruction performance. However, the proposed two-stage sensing approach has the advantage of a much simpler circuit implementation (with on-chip Bernoulli) compared to Sapiro's approach as explained previously.

Algorithm 1: Incremental Update of Off-chip Sensing Matrix $\mathbf{P}$

Input: $\mathbf{S}$, $\mathbf{D}_{new}$ and $\mathbf{U}$, $\mathbf{\Sigma}$ from the SVD of $\mathbf{SD}$
Output: The updated sensing matrix $\mathbf{P}$

$\tilde{\mathbf{D}}_{new} \leftarrow orth(\mathbf{SD}_{new} - \mathbf{U}\mathbf{U}^\top\mathbf{SD}_{new})$
$\mathbf{R} \leftarrow \begin{bmatrix} \mathbf{\Sigma} & \mathbf{U}^\top\mathbf{SD}_{new} \\ \mathbf{0} & \tilde{\mathbf{D}}_{new}(\mathbf{SD}_{new} - \mathbf{U}\mathbf{U}^\top\mathbf{SD}_{new}) \end{bmatrix}$
$\tilde{\mathbf{U}}, \tilde{\mathbf{\Sigma}} \leftarrow svd(\mathbf{R})$
$\mathbf{U}_{new} = [\mathbf{U}\ \tilde{\mathbf{D}}_{new}]\tilde{\mathbf{U}}$
$\mathbf{\Sigma}_{new} = \tilde{\mathbf{\Sigma}}$
return $\mathbf{P} = \mathbf{U}_{new}\mathbf{\Sigma}_{new}^{-1}\mathbf{U}_{new}^\top$

In some applications, neuroscientists need to perform longitudinal analysis to explicitly deal with slow changes of spike shapes.[52] In other cases, the sparsely firing neurons need special attentions.[53] Both circumstances require an update of the dictionary $\mathbf{D}$ and the corresponding off-chip sensing matrix $\mathbf{P}$ in our framework. The

dictionary update in our approach is simply concatenation, which gives $[\mathbf{D}\ \mathbf{D}_{new}]$, where $\mathbf{D}_{new} \in \mathbb{R}^{N \times K_{new}}$ is the new full acquisition of the neural signal. Since the off-chip sensing matrix $\mathbf{P}$ is related to the SVD of $\mathbf{SD}$ by (3.2.3), we can leverage the incremental PCA scheme[54] to update it efficiently. Our algorithm for the incremental update of sensing matrix $\mathbf{P}$ is presented in Algorithm 1. Here, $orth(\cdot)$ performs orthogonalization via QR and $svd(\cdot)$ performs SVD. The proposed algorithm has a computational complexity of $O(NK_{new}^2)$, versus $O(N(K + K_{new})^2)$ for recomputing the SVD using the whole new dictionary. Moreover, the total storage required reduces to $O(N(K + K_{new}))$, down from $O(N(K + K_{new})^2)$. To remove the old data from the dictionary and avoid the dictionary from growing too large, we use a forgetting factor to gradually remove them.

3.2.3 Restoration from Spike Segments

The spike detection devices only transmit the signal segments with spikes.[32–36] Our system includes a spike detection module, which was used in our previous work[43] to provide prior to guide the CS recovery of the full signal. In this section, we will show that using the spike detection module, we could have two new working modes, (i) spike detection with off-chip restoration to the full signal (Spike Restoration mode); and (ii) spike detection with on-chip CS and off-chip restoration to the full signal (Spike CS + Restoration mode). This enhances the flexibility of the system to obtain an all-in-one device. Moreover, the Spike CS + Restoration mode is the combination

of traditional Full Signal CS mode and Spike Restoration mode and can provide more aggressive compression performance.

3.2.3.1 Spike Restoration Mode

In this mode, only the spike segment $\mathbf{x}_\Omega$ is transmitted, where Ω indicates the time stamps of the spike segment. This changes our noiseless signal model into:

$$\mathbf{x}_\Omega = \mathbf{D}_\Omega \mathbf{a}, \tag{3.2.4}$$

where D_Ω is the sub-matrix built by extracting the corresponding rows in Ω from $\mathbf{D}$. This signal model is elaborated in Fig. 3.4. It is straightforward to see that if the full signal $\mathbf{x}$ could be sparsely represented by a few full dictionary atoms, so is its spike segment, but with the truncated dictionary atoms. Thus, if only the spike segment $\mathbf{x}_\Omega$ is transmitted off-chip, the full signal could be restored by first finding $\mathbf{a}$ with:

$$\text{Noisy:} \quad \min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{s.t.} \quad \|\mathbf{x}_\Omega - \mathbf{D}_\Omega \mathbf{a}\|_2 \leq \sigma. \tag{3.2.5}$$

After we figure out the sparse coefficients, the estimate of the full signal could be calculated by (2.2.3). Using only spike segment for sparse recovery does lead to inevitable loss of information and the sparse coefficient solution $\mathbf{a}$ could be different from the result using full signal. However, in practice, we could achieve good performance as shown later.

3.2.3.2 Spike CS + Restoration Mode

Built upon our Full Signal CS mode and the Spike Restoration mode, we would like to further explore the performance of the proposed CS-based design on spike segments. This mode could potentially lead to an even higher compression ratio than any other design because it essentially combines three compression elements into one framework − spike detection, on-chip CS and off-chip Puffer Transformation. In this case, we treat the spike segment as the signal and the measurement signal after the off-chip sensing stage becomes:

$$\mathbf{z} = \mathbf{PSx}_\Omega = \mathbf{PSD}_\Omega\mathbf{a}. \tag{3.2.6}$$

Notice that the number of columns in the sensing matrix $\mathbf{S}$ is no longer equal to the length of the full signal, but the size of spike segment. Similarly, the off-chip sensing matrix $\mathbf{P}$ is found using $\mathbf{D}_\Omega$ instead of $\mathbf{D}$. Here we still use the same notations for simplicity. The sparse coefficients are found by:

$$\text{Noisy:} \quad \min_{\mathbf{a}} \|\mathbf{a}\|_1 \quad \text{s.t.} \quad \|\mathbf{z} - \mathbf{PSD}_\Omega\mathbf{a}\|_2 \leq \sigma, \tag{3.2.7}$$

and the full signal is again restored using (2.2.3). Here, another benefit for Spike CS + Restoration mode is that even though we only take CS measurements of spike segments, the recovery results are the entire global signal containing the non-spike

segments as well.

3.2.4 Recovery Algorithm

In this section, we will present our CS recovery algorithm in the Tetrode system setup. Multi-channel recording systems leveraging the joint sparsity concept have been explored previously.[55–58] Our underlying joint sparsity model for Tetrode CS system is shown in Fig. 3.5. For each channel of Tetrode, we use different sparsifying dictionaries $\mathbf{D}$ (color coded in Fig. 3.5) to indicate differences in the shapes of the captured neural signal. However, since all four channels pick up the activity from the same neuron at the same time, there exists a strict matching between the four channels, which is the type of neurons detected. To capture this correlation, we enforce the corresponding sparse coefficients to choose the same support locations under the assumption that the atoms of four sparsifying dictionaries also align in time. For our data dictionary, we guarantee this alignment by choosing the pre-acquired data of different electrodes with the same time stamp and placing them into the same columns of the dictionaries for these electrodes.

This joint sparsity problem could be solved by greedy methods, optimization-based approaches or Bayesian inference techniques. For our framework, we use Bayesian inference for sparse modeling and choose the Spike and Slab prior,[59] which is a mixture of Gaussian distribution and direc delta function. Our previous work[60] have illustrated how to use the Spike and Slab model for hierarchical sparse modeling. Here,

we follow a similar methodology to develop the model for joint sparsity problem in (2.1.3). After incorporating our two-stage sensing process, the likelihood function for the joint sparsity model is as follows:

$$\text{Likelihood:} \quad \mathbf{Z}|\mathbf{P}_t, \mathbf{S}_t, \mathbf{D}_t, \mathbf{A}, \sigma^2 \sim \prod_{t=1}^{T} \mathcal{N}\left(\mathbf{P}_t\mathbf{S}_t\mathbf{D}_t\mathbf{a}_t, \sigma^2\mathbf{I}\right). \tag{3.2.8}$$

Here $\mathbf{Z}$ and $\mathbf{A}$ are the concatenation of $\mathbf{z}_t$ and $\mathbf{a}_t$, where $t = 1, ..., T$ represents each of the electrodes. For the case of Tetrodes, T is equal to 4, while $T = 1$ indicates a single electrode setup. For the Tetrode case, the dictionary for each electrode is unique, therefore the corresponding off-chip sensing matrix is also unique because of (3.2.3). The parameter σ is the noise standard deviation for the Gaussian likelihood. The prior distribution using Spike and Slab is:

$$\text{Prior:} \quad \mathbf{A}|\sigma_a^2, \boldsymbol{\gamma} \sim \prod_{t=1}^{T}\prod_{i=1}^{K} \gamma_i\mathcal{N}(0, \sigma_a^2) + (1 - \gamma_i)\delta, \tag{3.2.9}$$

where $\boldsymbol{\gamma}$ is the latent variable indicating the active support of sparse coefficients, and σ_a denotes the spread of the Slab part. It can seen that for Spike and Slab prior, the desired degree of sparsity is directly related to the weight $\boldsymbol{\gamma}$ assigned to the Slab part. We enforce measurements from different electrodes to share the same $\boldsymbol{\gamma}$, so that they will have the desired joint sparsity structure in the coefficient matrix $\mathbf{A}$. We have also shown in our previous work that Spike and Slab prior has a close connection with Elastic Net formulation,[61] resulting in a sparser solution while maintaining

the grouping characteristic. The interested reader could refer to[60] for more details. Finally, the hyperprior for the latent variable $\boldsymbol{\gamma}$ is:

$$\text{Hyperprior:} \quad \boldsymbol{\gamma}|\kappa \sim \prod_{i=1}^{K} \text{Bernoulli}(\kappa), \tag{3.2.10}$$

where κ controls the sparsity. In all our experiments, we fix the parameters σ, σ_a to be 1 and κ to be 0.1. Our results are not sensitive to the choice of these parameters.

Algorithm 2: EP algorithm of joint sparsity using Spike and Slab

Input: $\mathbf{P}_t$, $\mathbf{S}_t$, $\mathbf{D}_t$ $(t = 1, ..., T)$, σ, σ_a and κ
Output: The sparse coefficient matrix $\mathbf{A}$
Initialize all $\tilde{f}_c$ terms (for $c = 1, 2, 3$) and $\mathcal{Q}$ to be non-informative.
while *any of the* $\tilde{f}_c$ *terms does not converge* **do**
    To refine each $\tilde{f}_c$ term, first find $\mathcal{Q}^c$ by dividing $\mathcal{Q}$ with $\tilde{f}_c$.
    Minimize $D_{KL}(f_c\mathcal{Q}^c||\tilde{f}_c\mathcal{Q}^c)$ to modify each of $m^c_{i,t}$, $v^c_{i,t}$, and p^c_i (for $i = 1, ..., K$, $t = 1, ..., T$ and $c = 1, 2, 3$).
    Find $\mathcal{Q}$ as the product of the new $\tilde{f}_c$ and $\mathcal{Q}^c$ to update $m_{i,t}$, $v_{i,t}$ and p_i (for $i = 1, ..., K$ and $t = 1, ..., T$).

Bayesian inference could be computationally demanding when using Spike and Slab priors. Thus, we choose an approximation method − expectation propagation (EP)[62] because our modeling only involves distributions from the exponential family and only the moments need to be updated. We represent the likelihood function (3.2.8), the prior for sparse coefficients (3.2.9) and the hyperprior for the latent variable (3.2.10) as different terms f_1, f_2 and f_3. Thus, the joint posterior distribution $\mathcal{P}(\mathbf{A}, \boldsymbol{\gamma}, \mathbf{Z}|\mathbf{P}_t, \mathbf{S}_t, \mathbf{D}_t)$ can be written as the product of these terms. We approximate

the posterior with following exponential family distribution:

$$\mathcal{Q} = \prod_{t=1}^{T}\prod_{i=1}^{K}\mathcal{N}(a_{i,t}|m_{i,t}, v_{i,t})\text{Bernoulli}(\gamma_i|p_i) \tag{3.2.11}$$

where $m_{i,t}$, $v_{i,t}$ (for $i = 1, ..., K$ and $t = 1, ..., T$) and p_i (for $i = 1, ..., K$) are the parameters to infer and will be our estimate of the mean and variance for sparse coefficients $\mathbf{A}_{i,t}$ (i, t-th element of $\mathbf{A}$) and mean for the latent variable $\boldsymbol{\gamma}$, respectively. Note that $m_{i,t}$, $v_{i,t}$ is specific for each $\mathbf{A}_{i,t}$, while p_i is the same for each row of $\mathbf{A}$ to favor the joint sparsity. The function f_1, f_2 and f_3 are also approximated with exponential family distributions as:

$$\tilde{f}_1 = z_1 \prod_{t=1}^{T}\prod_{i=1}^{K}\mathcal{N}(a_{i,t}|m^1_{i,t}, v^1_{i,t}) \tag{3.2.12}$$

$$\tilde{f}_2 = z_2 \prod_{t=1}^{T}\prod_{i=1}^{K}\mathcal{N}(a_{i,t}|m^2_{i,t}, v^2_{i,t})\text{Bernoulli}(\gamma_i|p^2_i) \tag{3.2.13}$$

$$\tilde{f}_3 = z_3 \prod_{i=1}^{K}\text{Bernoulli}(\gamma_i|p^3_i). \tag{3.2.14}$$

Here $m^1_{i,t}$, $v^1_{i,t}$, $m^2_{i,t}$, $v^2_{i,t}$ (for $i = 1, ..., K$ and $t = 1, ..., T$) and p^2_i and p^3_i (for $i = 1, ..., K$) are the intermediate parameters to be updated in each EP update, whereas z_1, z_2 and z_3 are normalization parameters. The complete EP procedure is shown in Algorithm 2. This algorihm could be applied to both the case of single electrode and Tetrode. For the detailed update procedures for the moments of the exponential

family distributions, readers can refer to.[62]

3.3 Experiment validation

In this section, we first demonstrate the recovery performances of the proposed dictionary, two-stage sensing method, and our approaches of restoring full signal from spike segments as well as CS of spike segments. Next, we demonstrate the advantage of our whole framework in terms of both reconstruction and classification performances. Finally, we show results of our Tetrode CS recovery versus recovering each electrode individually. The recovery results are based on MATLAB simulation. Since the CS circuits are implemented using digital circuits which does not add additional noise to the signal, its behavior can be precisely modeled using MATLAB simulation.

Both synthetic and real datasets are employed in various experiments. We use the Leicester neural signal database,[47] which contains 20 simulation datasets. Each dataset contains spikes from three different types of neurons with different noise levels. The datasets are named by the difficulty to perform spike sorting, such as Leicester Difficult1, Difficult2, Easy1, and Easy2. We also carry out benchmarking on the publicly available dataset hc-1,[63] which is the recording from nearby neurons in the hippocampus of an anesthetized rat. We take 128 and 64 samples around each spike to form the signal frame for Leicester datasets and hc-1 dataset, respectively. To simplify the comparison, we retain the signal containing only one spike, while the case

with multiple spikes in one frame is addressed in our previous work.[43] All experiments are ran 10 times with average results being reported. Our result is consistent among both synthetic and real datasets.

3.3.1 Performance of the Proposed Dictionary

Under different compression ratios $CR = \frac{M}{N}$, we compare four different choices of dictionaries in the CS framework, including the proposed data dictionary, trained dictionary,[43] wavelet dictionary,[2] and Gabor dictionary.[58] We also include the spike detection[34] for comparison. We choose random Bernoulli matrix for all CS-based approaches. To accommodate the need for training, we randomly split the data into two halves with equal sizes, with one part for training and the other part for testing. The parameters for dictionary learning is the same as in.[43] The result is found in term of Signal to Noise and Distortion Ratio ($SNDR$),[2] which is defined as:

$$SNDR = 20 \log \frac{||\mathbf{x}||_2}{||\mathbf{x} - \hat{\mathbf{x}}||_2}. \tag{3.3.1}$$

We employ Algorithm 2 to solve (2.1.2) and recover the signal $\hat{\mathbf{x}}$ by (2.2.3). The results for datasets Leicester - Difficult1 and Leicester - Easy1 with 0.005 noise std are shown in Fig. 3.6. We can see that the proposed approach using data as the dictionary works comparably as using the trained dictionary and far better than other approaches. Compared to the trained dictionary, the proposed data dictionary

is much simpler to update, hence more suitable for large scale monitoring applications.

3.3.2 Performance of the Proposed Two-Stage Sensing

Under different compression ratios CR, we also compare four different choices of sensing matrices $\mathbf{S}$ in the CS framework, including the proposed two-stage sensing, Sapiro's approach,[15] random Bernoulli matrix[43] and random Gaussian matrix. For a fair comparison, we use data dictionary for all methods. Each time, the data is randomly splitted into two halves with one for training and the other for testing. The rest of the experiment setup is the same as in previous section. The results for datasets Leicester - Difficult2 and Leicester - Easy2 are shown in Fig. 3.7. As expected, the proposed two-stage sensing approach is worse than the Sapiro's approach, but it has the strength of an efficient circuit implementation. Meanwhile, the proposed two-stage CS approach is more than 2dB better than random matrices when CR is above 15%, which we use as the empirical threshold. Note that we only perform two-stage sensing when the CR is above this threshold to achieve the overall best performance.

3.3.3 Restoration from Spike Segments

We use Leicester - Difficult1, Difficult2, Easy1 and Easy2 datasets to demonstrate the performance of our proposed approaches of restoring full signal from spike seg-

Table 3.1: Performance of the Proposed Spike Restoration Mode and Spike CS + Restoration Mode in *SNDR*.

Spike Restoration with Spike Size of 13				
Datasets	Easy1	Easy2	Difficult1	Difficult2
Effective $CR = 10\%$	11.3	11.4	10.7	10.9
Spike CS Recovery + Restoration with Spike Size of 13				
Datasets	Easy1	Easy2	Difficult1	Difficult2
Effective $CR = 2\%$	9.6	9.8	7.8	9.6
Effective $CR = 4\%$	10.8	10.8	10.0	10.3
Effective $CR = 6\%$	11.0	11.0	10.3	10.6

ment. The size of the spike segment is fixed to 13, corresponding to a CR of 10%. The data is also randomly divided into two equal sets for training and testing, respectively. We keep the full signal for training data while truncating the test data by spike detection algorithm. We then test the Spike Restoration Mode by just transmitting the spike segments and use the trained dictionary to solve (3.2.5) and (2.2.3) with the results shown in Table 3.1. The proposed method could achieve a *SNDR* above 10dB at the CR of 10%.

Next, we fix the size of the spike segment to 13, and test the performance of our proposed approach of restoring full signal from CS measurements of spike segment. We vary the CR of spike segments between 20%, 40% and 60%, which gives an effective CR of 2%, 4%, and 6%, respectively. Then we test the Spike CS + Restoration Mode by transmitting the CS measurements of spike segments and use the trained dictionary to solve (3.2.7) and (2.2.3). The reconstruction performance of the same

datasets are shown in Table 3.1. Surprisingly, we could achive almost 10dB *SNDR* for all datasets at an effective CR of merely 2%. There is an inevitable loss of *SNDR* for Spike CS mode due to the lossy nature of CS, but the degradation level is negligible. Even though we further reduce the number of measurements (effective CR from 6% to 2%), the recovery performance is not changing much. This confirms our hypothesis that the neural signal can be very sparsely represented using the proposed data dictionary meaning that the number of measurements M required could be reduced significantly.

To understand the results more intuitively, two examples of the reconstruction of the same signal for Spike Restoration mode ($CR = 10\%$) and Spike CS Recovery + Restoration mode ($CR = 2\%$) are shown in Fig. 3.8. For Spike Restoration mode, we can get the perfect alignment of the spike segment while it is slightly mis-aligned in the Spike CS Recovery + Restoration mode due to the CS compression. However, we can see that as long as we capture the spike segment of neural signal, we could capture the main features of the data and therefore achieve a much higher compression ratio than CS of the full signal.

3.3.4 Performance of Overall Framework for Single Electrode

Now we are ready to compare the overall framework, including the Full Signal CS mode, Spike Restoration Mode and Spike CS + Restoration mode with Signal Dependent Neural Compressed Sensing Method (SDNCS),[43] DWT based CS method (DWT-CS),[2] Transformation based method (on-chip DWT),[37] and Spike detection method.[34] We randomly split the data into 20% for training and 80% for testing. The reconstruction and classification results for Leicester - Difficult1 and Easy2 with 0.005 noise std are shown in Fig. 3.9. We use the same wavelet based classifier (WLC) as in.[43] It can be seen that our proposed Full Signal CS mode consistently yields approximately 5dB gain than SDNCS because our Bayesian inference algorithm and the two-stage sensing approach are more effective in regularizing the optimization problem. Our Spike Restoration Mode and Spike CS + Restoration mode also outperforms spike detection method significantly for both reconstruction and classification. All three modes of our system can achieve above 95% classification accuracy at the CR around 6%. Notice that our three working modes outperform the on-chip DWT approach when the measurement number M is smaller than 8 ($CR < 6\%$), so it provides a better solution which yields better reconstruction and classification performance with much simpler on-chip implementation. Interestingly, the Spike Restoration Mode and Spike CS + Restoration mode can achieve higher clas-

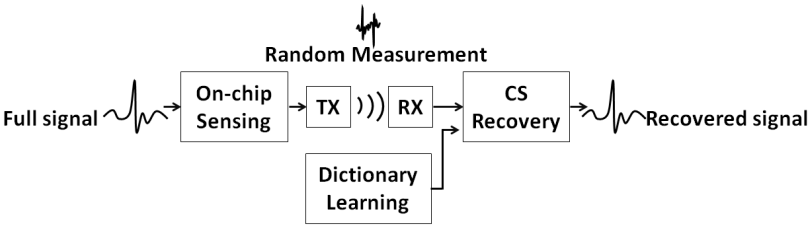
sification accuracy than our Full Signal CS mode. This is because CS is essentially a lossy compression method and the information loss in the CS compression is more significant for full signal than the spike component because the CR has to be higher to achieve the same number of measurements. Although we lose the information on the non-spike segment for Spike Restoration Mode and Spike CS + Restoration mode, this is in some extent compensated by our full signal dictionary.

We also use hc-1 dataset to compare different approaches. The size of the spike segment is chosen to be 13 for a signal length of 64. For Spike CS + Restoration mode, we further compress the spike segment to a measurement of size 6, which yields an effective CR to be 10%. An example of the reconstructed signal by different approaches are shown in Fig. 3.10 together with the corresponding *SNDR*. Our new framework of Full Signal CS mode outperforms our previous design −− SDNCS. And the two modes based on spike segments could achieve a *SNDR* around 10dB with 10% CR. Notice that the performance comparison of the two modes using spike segment is related to the size of the spike segments and the characteristics of the sparsifying dictionary after truncation, thus it is hard to tell which one could give better results in general.

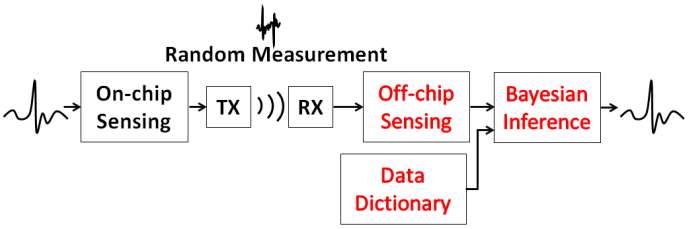
3.3.5 Tetrode CS

Previous experiments only consider the single electrode setup, while our proposed algorithm could take into account the correlation between four electrodes for Tetrode

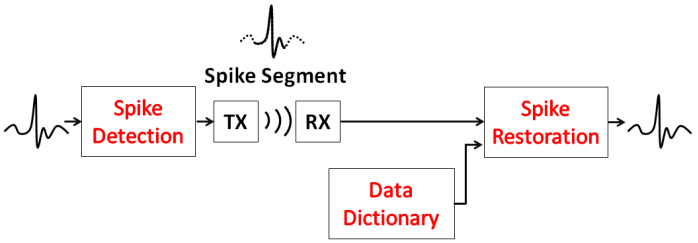
monitoring. Using hc-1 dataset, we compare the performance of Tetrode CS reconstruction using joint sparsity versus reconstructing each electrode independently. We also compare using data dictionary with Gabor dictionary.[58] To emphasize the main factors of joint sparsity and choices of dictionary, we use our Bayesian inference algorithm for all cases with the only difference in T, which will enforce joint sparsity when there are multiple electrodes. After pre-processing (i.e., bandpass filter), we divide the hc-1 dataset using a frame size of 64 ($N = 64$). Half of the dataset for each electrode is used as the data dictionary and the remaining for testing. Note that the time stamps of the signal used for training are the same for all electrodes, which is the key assumption for using joint sparsity in Tetrode CS. The experiment is again performed 10 times with different random partition each time and the average result is reported as in Fig. 3.11. We could see that the proposed Tetrode CS technique using joint sparsity and data dictionary consistently outperforms the individual CS reconstruction using data dictionary by about 1dB. Meanwhile, the approaches using data dictionary consistently outperform ones using Gabor dictionary, which is similar as the results in Fig. 3.9.



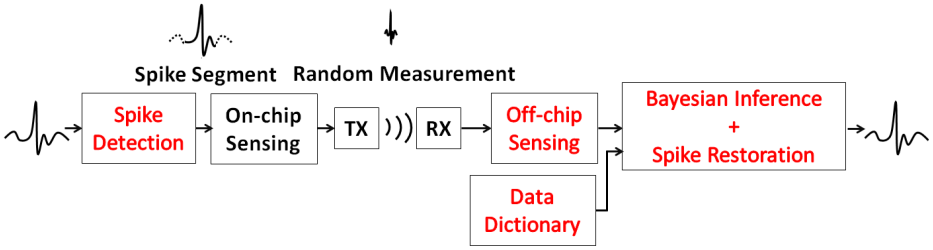
(a) Our previous design[43]



(b) Proposed system - Full Signal CS mode



(c) Proposed system - Spike Restoration mode



(d) Proposed system - Spike CS + Restoration mode

Figure 3.1: Comparison of our previous design and the three working modes of the proposed system. The new design elements are highlighted in red.

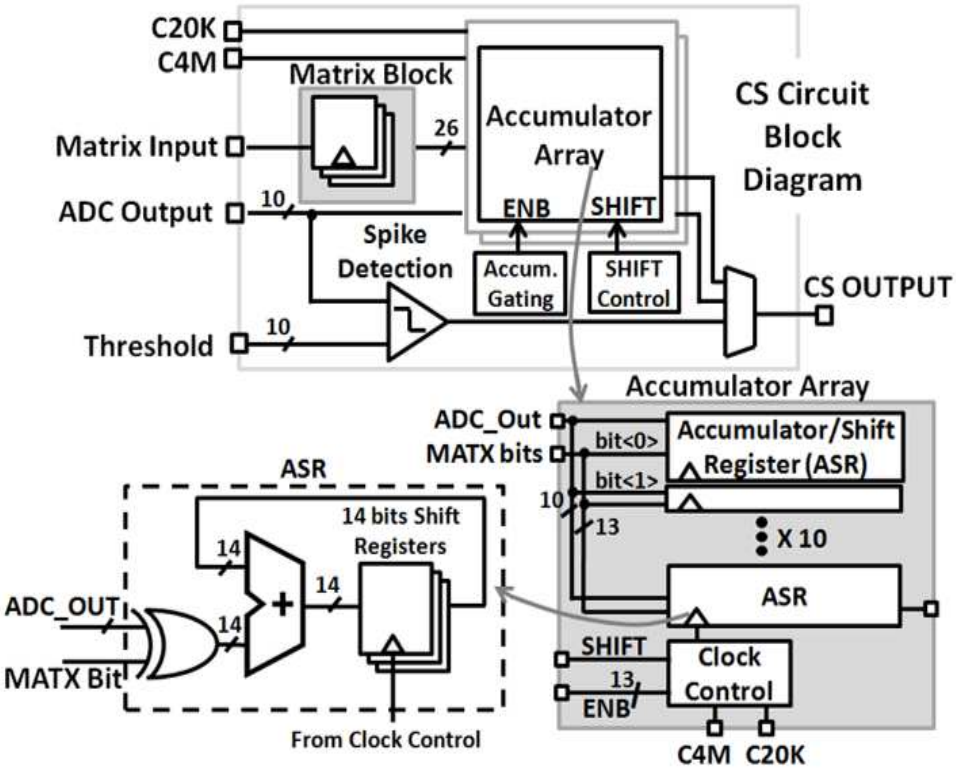


Figure 3.2: Architecture of our CS circuit.

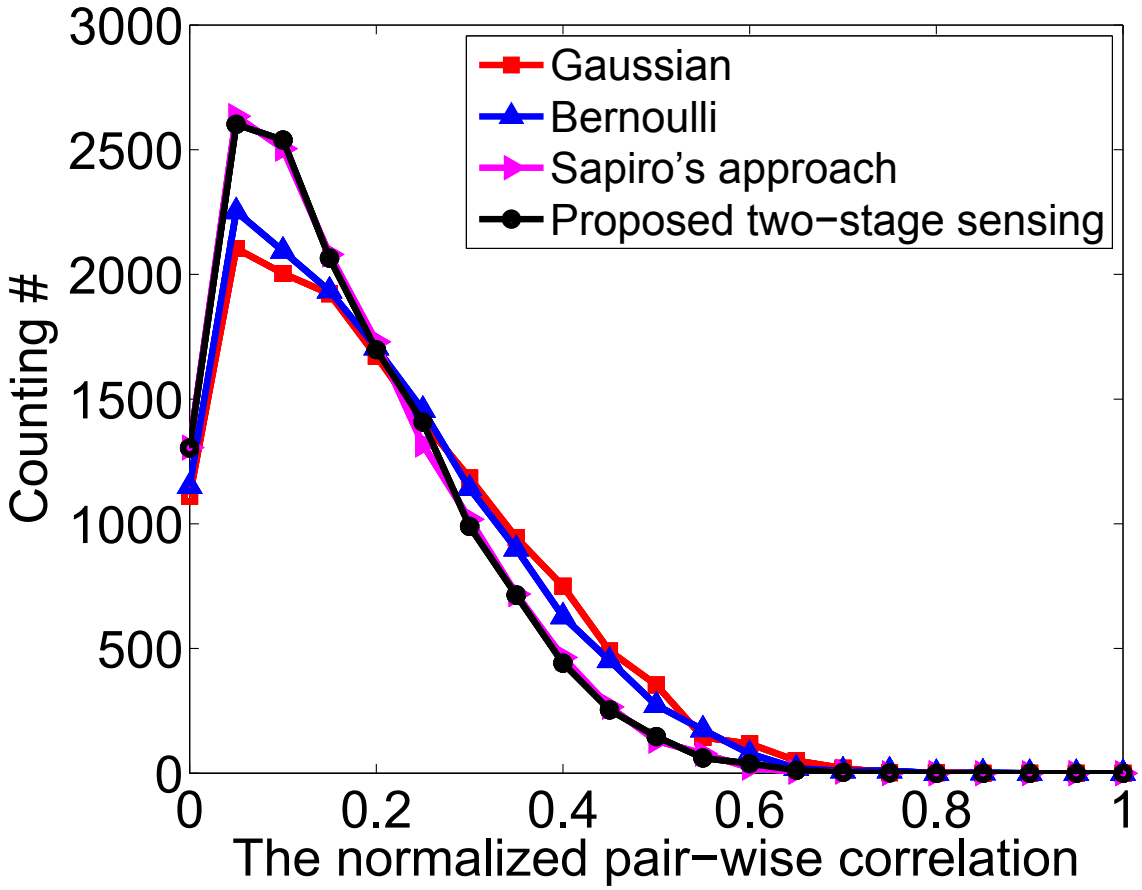


Figure 3.3: Comparison between the proposed two-stage sensing method and other sensing methods

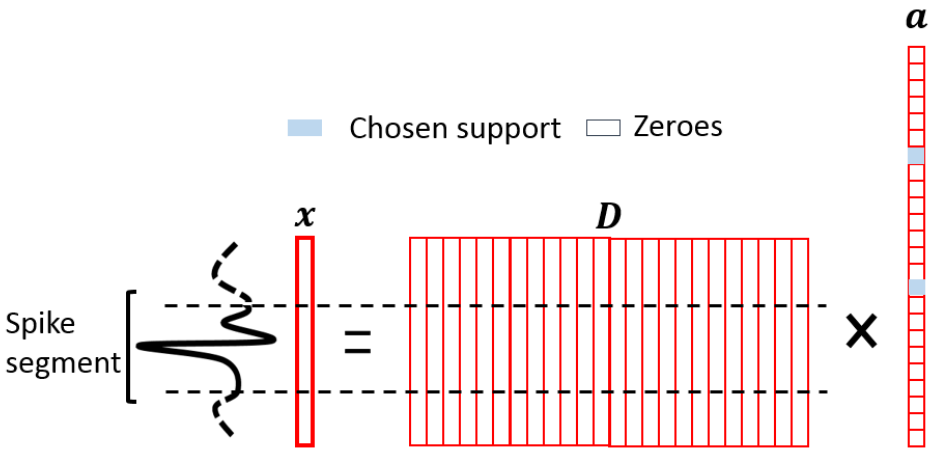


Figure 3.4: The signal model for Spike Restoration mode. The black dotted line represents the time stamps of the spike segment. Although the spike segment is truncated from the full signal, it can still be captured by a sparse representation with respect to the truncated dictionary.

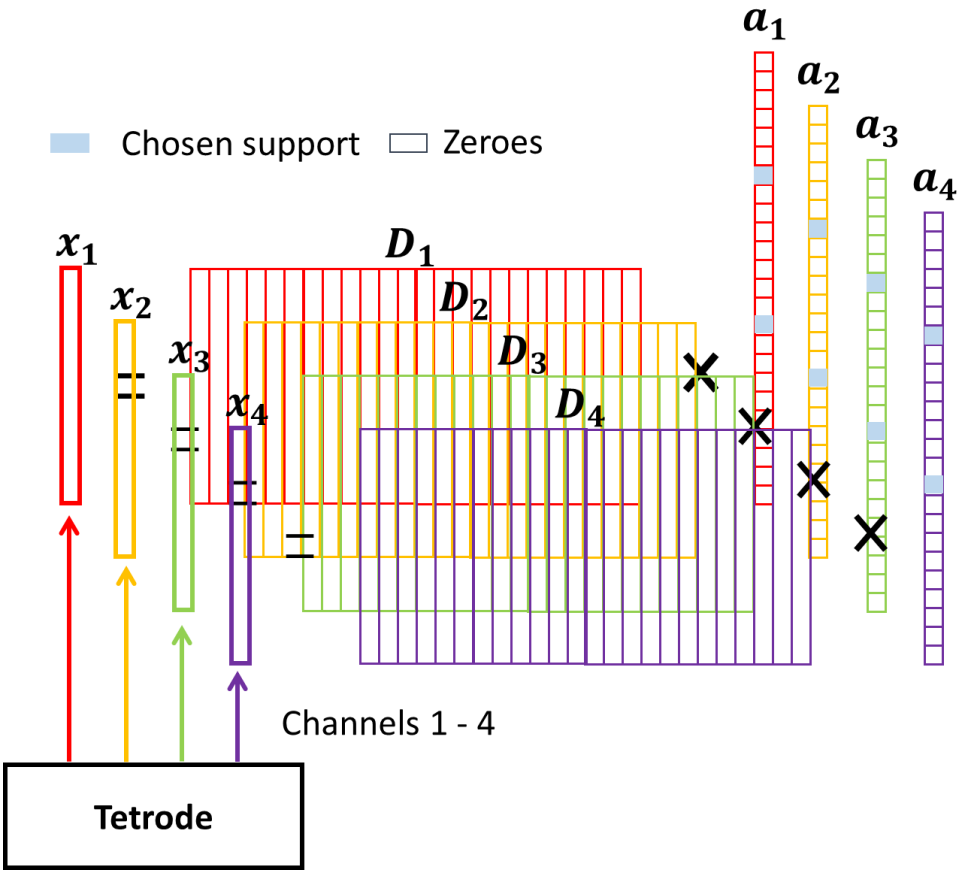
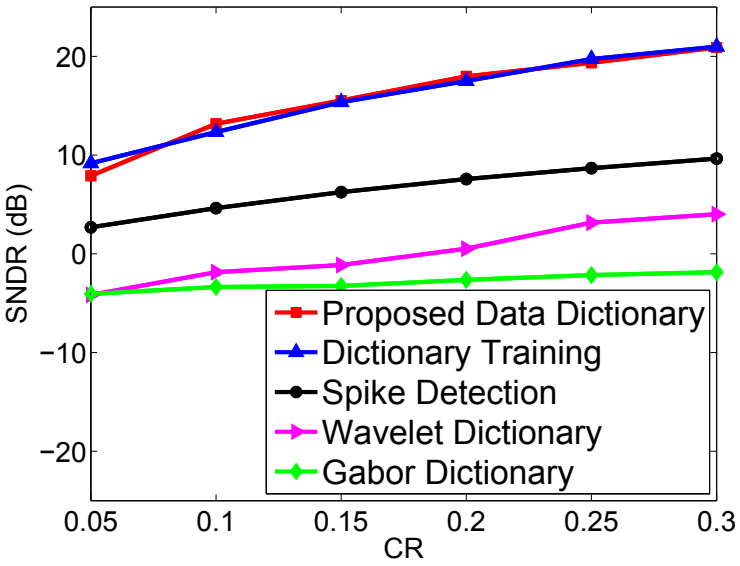
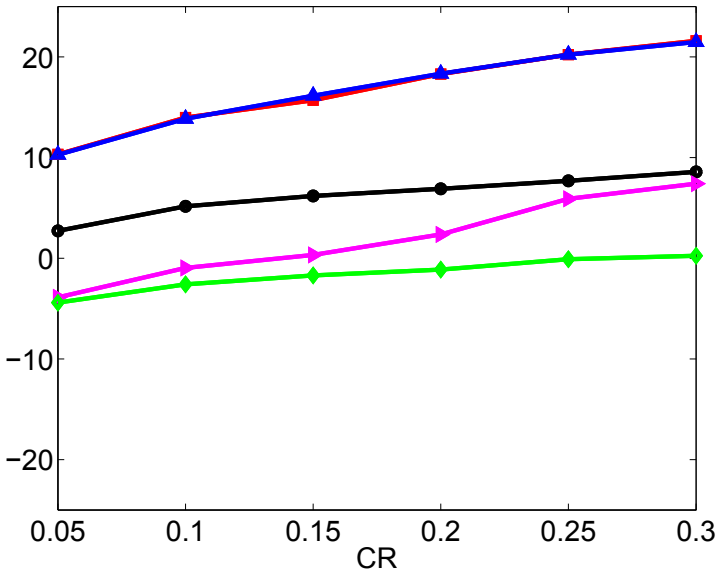


Figure 3.5: The underlying joint sparsity model for Tetrode CS. Notice that the support of the sparse coefficients for different channels are the same given that the dictionary atoms for different electrodes are aligned in time.

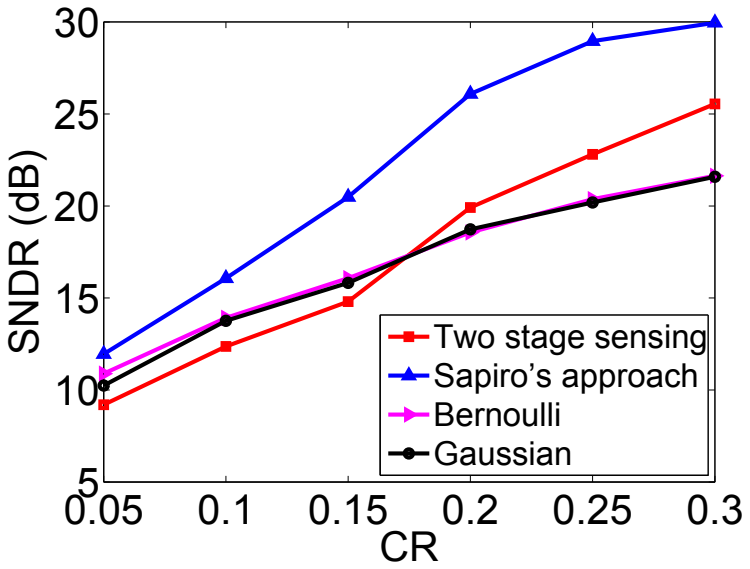


(a) Leicester - Difficult1 dataset

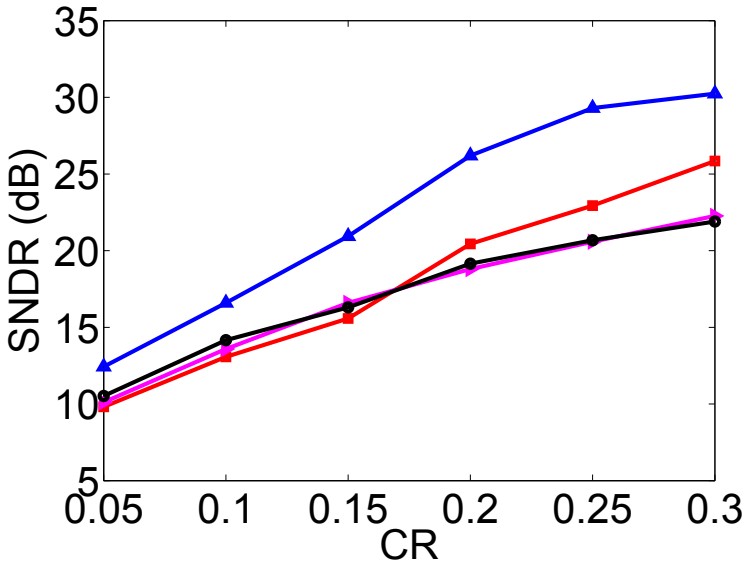


(b) Leicester - Easy1 dataset

Figure 3.6: Comparison of different sparsifying dictionaries.

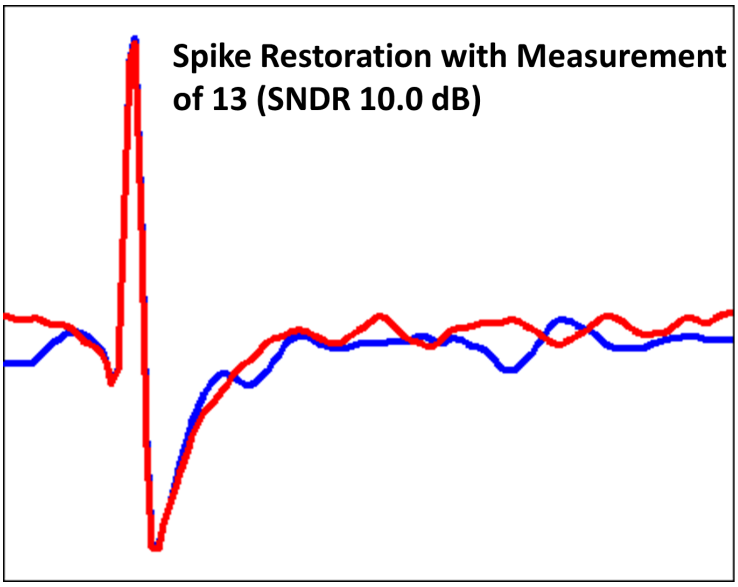


(a) Leicester - Difficult2 dataset

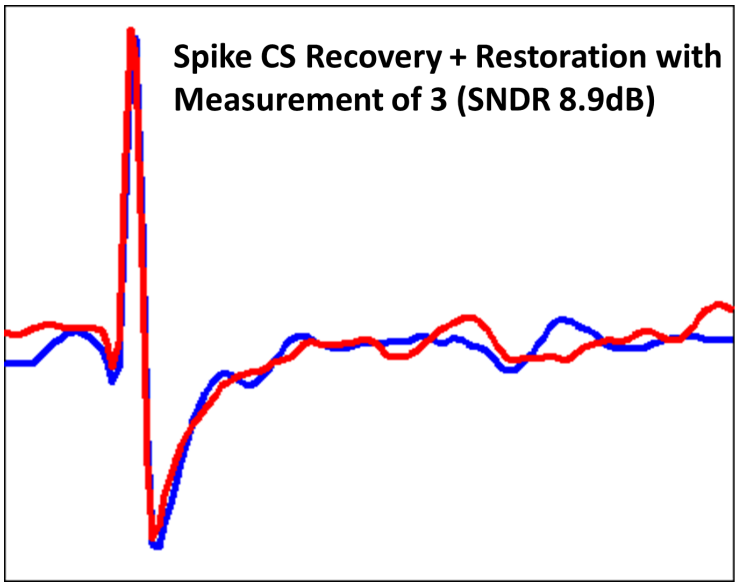


(b) Leicester - Easy2 dataset

Figure 3.7: Comparison of different sensing matrices.



(a) Spike Restoration mode



(b) Spike CS Recovery + Restoration mode

Figure 3.8: Example results of Spike Restoration mode and Spike CS Recovery + Restoration mode for same neural signal from Leicester - Easy2 dataset. Red indicates the reconstruction results and blue represents the ground truth. The corresponding *SNDR*s for this specific signal are also included.

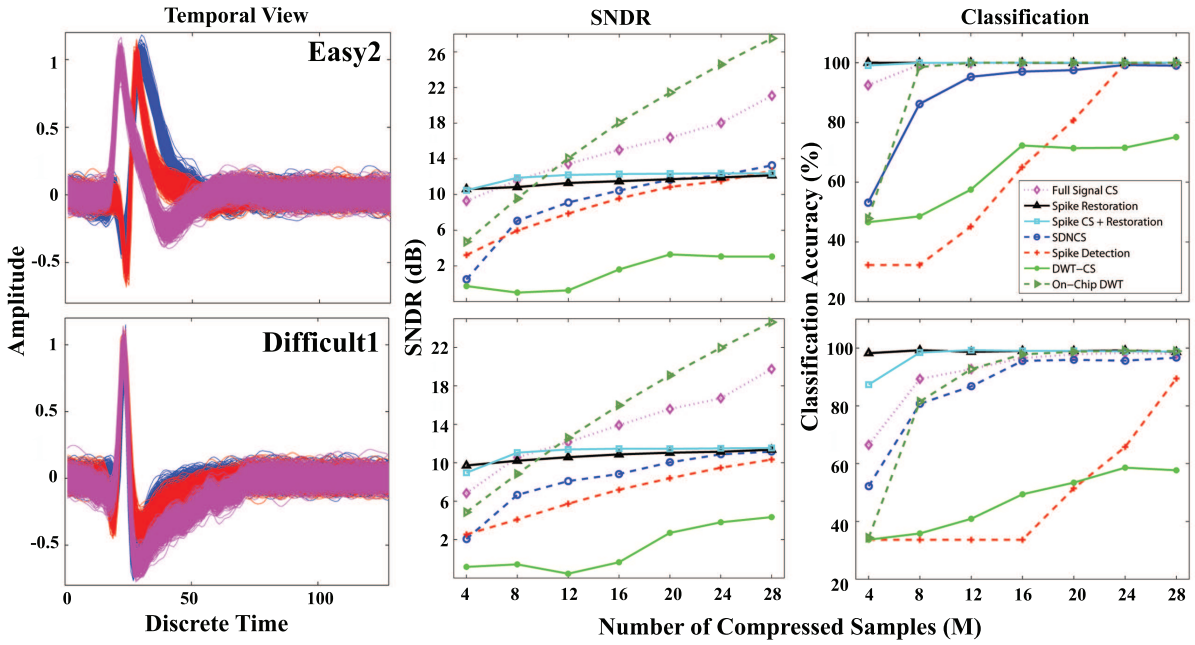
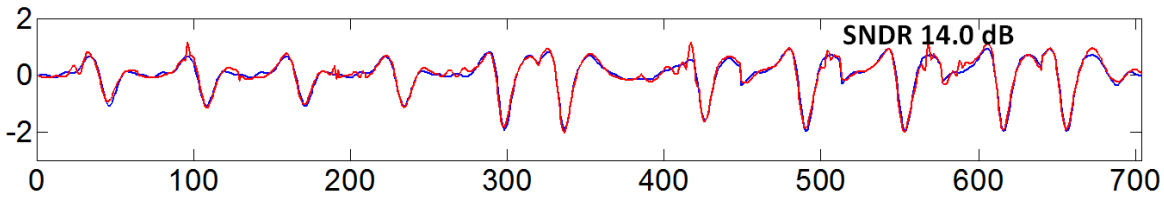
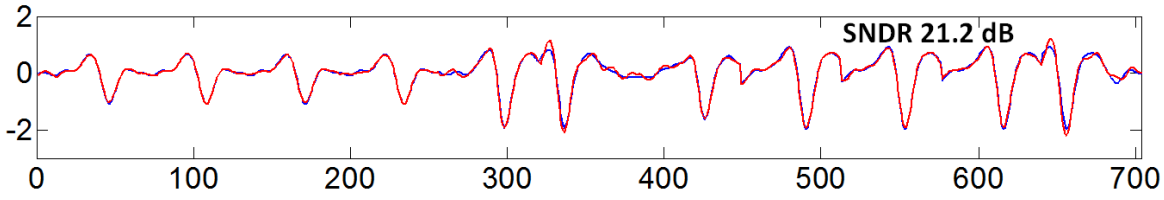


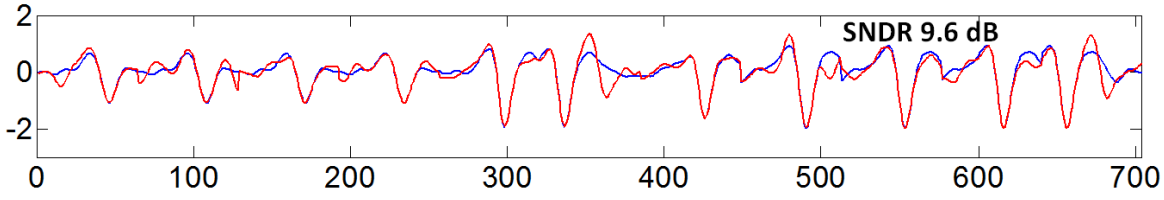
Figure 3.9: Temporal views of the test signal(left column), recovery *SNDR* (middle column) and classification accuracy (right column) comparison of on-chip DWT (dark green triangulated traces), Spike Detection (red traces), DWT-CS (green dotted traces), SDNCS (blue traces), our Full CS mode (purple trace), our Spike Restoration mode (black trace) and our Spike CS + Restoration mode (cyan trace). In the plot of recovery *SNDR*, the window is set to display values in the range of 0 to 26 dB. In the plot of classification accuracy, the window is set to display values in the range of 20% to 100%.



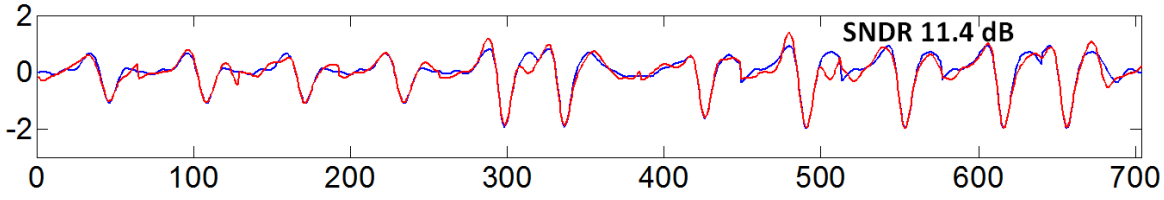
(a) Our previous design[43] with CR of 20%



(b) Proposed system - Full Signal CS mode with CR of 20%



(c) Proposed system - Spike Restoration mode with CR of 10%



(d) Proposed system - Spike CS + Restoration mode with CR of 10%

Figure 3.10: An example of the reconstruction results on hc-1 dataset. Blue represents the original signal and red indicates the reconstruction results by each approach.

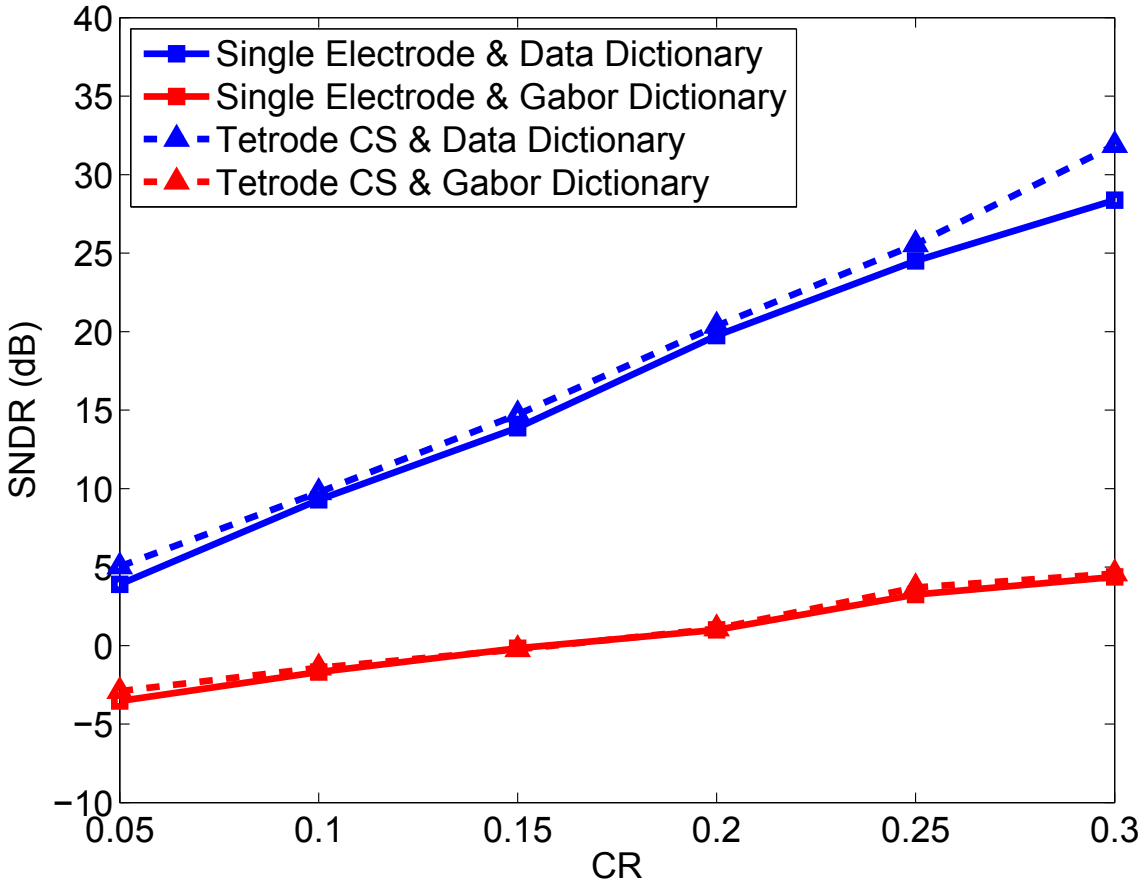


Figure 3.11: Comparisons of Tetrode CS recovery using single electrode approach versus joint sparsity approach, and data dictionary versus Gabor dictionary.

Chapter 4

Structured Dictionary Learning for Classification

4.1 Introduction

Traditionally, dictionaries are designed for desired properties in spatial or frequency domain or both.[4] Recently, a different methodology to learn the dictionary from data is shown to capture data characteristics better. There are two directions for designing such a signal dependent dictionary:

(i) Using data directly as the dictionary: Wright *et al.*[48] proposed a sparse representation-based classifier (SRC) that concatenates the training data from different classes into a single dictionary and uses class-specific residue for face recognition. Besides supervised tasks, a data dictionary is also utilized to cluster the high dimensional data by

finding intrinsic low-dimensional structures with respect to itself.[64]

(ii) Training a dictionary using data: Training a dictionary such that the data could be sparsely represented with has demonstrated its advantages in image processing tasks.[16,65,66] Yu *et al.*[21] justified that encoding data with dictionary atoms in its neighborhood can guarantee a nonlinear function of the data (i.e., a Lipschitz smooth function as defined in[21]) to be well approximated by a linear function with this local coordinate coding.

In contrast to the former approach, the learned dictionary in the latter approach removes the redundant information in the learning process, therefore the size of the dictionary does not grow with the size of the data. In this paper, we will focus on the latter approach. Moreover, we assume that the data has been properly aligned, although data alignment[67,68] is another active research area with growing interests.

4.1.1 Dictionary Learning for Reconstruction

Dictionary learning (DL) is first attempted for the purpose of reconstruction. The learning process can be described by the following optimization problem:

$$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1||\mathbf{a}_i||_q).$$

Given training data $\mathbf{x}_i \in \mathbb{R}^M$ $(i = 1, ..., N)$, the dictionary $\mathbf{D} \in \mathbb{R}^{M\times K}$ and corresponding sparse coefficients $\mathbf{A} \in \mathbb{R}^{K\times N}$ are both learned. Each column of $\mathbf{D}$ and $\mathbf{A}$

are denoted as $\mathbf{d}_j$ $(j = 1, ..., K)$ and $\mathbf{a}_i$ $(i = 1, ..., N)$, respectively. The dictionary size K is typically larger than signal dimension M. The ℓ_q-norm is chosen to promote sparsity while the trade-off between data fidelity and the sparsity regularization is balanced by tuning the parameter λ_1.

This non-convex optimization problem is usually solved by iterating between sparse coding and dictionary updating. In the sparse coding stage, the sparse coefficient $\mathbf{a}_i$ is found with respect to a fixed dictionary $\mathbf{D}$. This can be carried out by greedy pursuit enforcing constraints on ℓ_0-norm,[16] convex optimization targeting ℓ_1-norm,[19,20] ℓ_2-norm minimization with locality constraint,[21] structured sparsity optimization[22,23] or Bayesian methods.[24] In the dictionary updating stage, each dictionary atom $\mathbf{d}_j$ is updated using only data with non-zero sparse coefficients on index j. This sub-problem can be solved by either block coordinate descent[19] or singular value decomposition.[16] Desirable features, such as multi-resolution[69] and transformation invariant,[70] could also be integrated to further improve performances in specific applications. All the dictionary atoms have same unit ℓ_2-norms to prevent some sparse coefficients from having very large or small values.

4.1.2 Dictionary Learning for Classification

We could treat sparse coefficients as features. In early works, dictionaries are learned in a reconstructive setup and then the corresponding sparse coefficients are used for classification tasks.[71,72] Recently, researchers start to explore ways to cus-

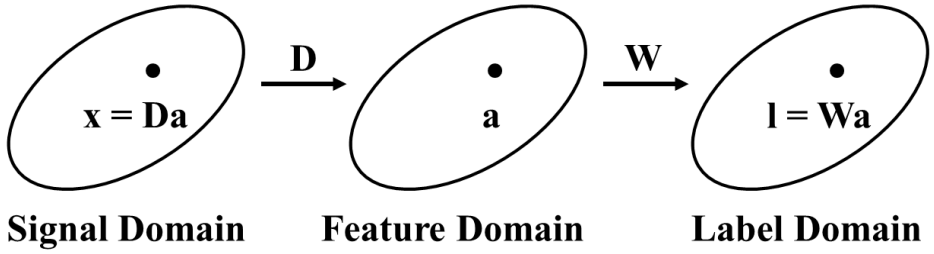


Figure 4.1: A schematic of using DL for classification.

tomize DL for classification. A general framework for this purpose is illustrated in Fig 4.1. The low dimensional signal $\mathbf{x}$ is mapped to its high dimensional feature (sparse coefficient) $\mathbf{a}$ using a learned dictionary $\mathbf{D}$, which could make the hidden patterns more prominent to capture. A classifier $\mathbf{W}$ is then utilized to predict the label vector $\mathbf{l}$. The key here is to promote $\mathbf{D}$ and $\mathbf{A}$ to be discriminative by adding extra constraints $f_{\mathbf{A}}(\cdot)$ and $f_{\mathbf{D}}(\cdot)$. Now the optimization problem becomes:

$$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1||\mathbf{a}_i||_q) + \lambda_2 f_{\mathbf{A}}(\mathbf{A}) + \lambda_3 f_{\mathbf{D}}(\mathbf{D}).$$

The function $f_{\mathbf{A}}(\cdot)$ could be a logistic function,[26] a linear classifier,[27,73] a label consistency term,[28,74] a low rank constraint[75] or Fisher discrimination criterion.[29] The choices of $f_{\mathbf{D}}(\cdot)$ could be to force either the sub-dictionaries for different classes,[30,76] or the atoms in each sub-dictionary to be as incoherent as possible.[77] The label can be assigned using class-specific residue[30] or linear classification.[28]

4.1.3 Our Contributions

Most methods mentioned in Section 4.1.2 adopt ℓ_0-norm or ℓ_1-norm based sparsity regularization, and add explicit constraints like a linear classifier into the DL formulation. This consolidation of DL and classifier into one objective function limits the freedom of choosing different classifiers and could complicate the optimization procedure of DL.[29] Different from these approaches, we improve the intrinsic discriminative properties of the dictionary by exploiting structured sparsity regularization. We introduce Hierarchical Dictionary Learning (HiDL) and Group Structured Dirty Dictionary Learning framework (GDDL) that incorporate structured sparsity on different levels. Our specific contributions are listed below.

- Different from approaches using group sparsity,[77] structured low rank[75] and hierarchical tree sparsity constraints[22] in DL, we propose to use hierarchical group sparsity in conjunction with the Dirty Model.[31] This way we can uniquely incorporate sparsity, group structure and locality in a single formulation, which are all desired features for an ideal dictionary to be used in classification. In contrast to the approaches that add extra constraints,[27,28] our formulation does not increase the size of the problem because the regularization is enforced implicitly.

- We show theoretically that our HiDL approach (GDDL in a single task setup) guarantees a perfect block structure of the sparse coefficients at the cost of a stricter condition, which is desired for classification problems. We also point

out that the condition is more likely to be satisfied when the dictionary size is smaller, thus making our method more favorable than ℓ_1-norm based DL.

- We employ both synthetic and real-world datasets to illustrate the superior performances of the proposed HiDL and GDDL. Meanwhile, we also point out scenarios where limitations still exist.

The chapter is organized as follows. In Section 4.2, we present our HiDL approach, extend it to learn dictionary from compressed measurements −− HiDL-CS, and generalize it to a multi-task setup −− GDDL. In Section 4.3, we use HiDL to derive conditions to guarantee its classification performance in a noiseless model.

4.2 Hierarchical and Group Structured Dirty Dictionary Learning For Classification

4.2.1 Motivation from a Coding Perspective

The coding stage in the DL process typically adopts ℓ_0- or ℓ_1-norm to encourage sparsity (the latter one is also referred as Lasso[7]). Its formulation is

$$\min_{\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1||\mathbf{a}_i||_1). \tag{4.2.1}$$

The corresponding prior distribution for Lasso is a multivariate Laplacian distribution with the independence assumption. Thus, the support is chosen largely dependent on the algorithm and the regularization parameter.

Since sparsity alone could not regulate the support location, locality-constrained linear coding (LLC)[78] is proposed to enforce locality instead of sparsity. The objective function of LLC is defined as:

$$\min_{\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1||\mathbf{e}_i \odot \mathbf{a}_i||_2^2), \tag{4.2.2}$$

where $\odot$ denotes the element-wise multiplication, and $\mathbf{e}_i \in \mathbb{R}^K$ is a weight vector indicating the similarity between signal and dictionary atoms. Locality constraint could lead to sparsity by controlling the size of the neighborhood. Conceptually, LLC endorses the local structure but loses the global perspective. For instance, the data lying on the class boundary could be coded with dictionary atoms from either side or both sides, creating ambiguity for classification tasks.

To promote both sparsity and group structure, Hierarchical Lasso (HiLasso)[10] is proposed as:

$$\min_{\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1 \sum_{g \in \mathcal{G}} ||\mathbf{a}_{i,[g]}||_2 + \lambda_2||\mathbf{a}_i||_1), \tag{4.2.3}$$

where $\mathcal{G}$ is a predefined group structure, and $\mathbf{a}_{i,[g]}$ is the sub-vector extracted from $\mathbf{a}_i$ using indices in group g. The group structure of HiLasso yields locality because similar

tasks $\mathbf{x}_i$ will select dictionary atoms corresponding to the same group g. Therefore for classification tasks, every group of dictionary atoms naturally inherits the label of the corresponding training data. In other words, the dictionary $\mathbf{D}$ can be regarded as the concatenation of sub-dictionaries $\mathbf{D}_1, ..., \mathbf{D}_C$ belonging to different classes, where C is the total number of classes and $\mathbf{D}_c$ has size K_c. In contrast to LLC, HiLasso captures the global information embedded in the group structure.

In the multi-task setup, different tasks are forced to share the same sets of dictionary atoms, which leads to a variant of HiLasso, called Collaborative HiLasso (C-HiLasso).[10] Although C-HiLasso captures the group correlation, it does not reveal explicitly if any atoms are shared by all tasks (within-class similarity) or uniquely utilized by individual task (within-class variation). The within-class variation makes the data clusters less compact and harder to classify. Therefore, it will be beneficial to separate it from the within-class similarity component to better capture the core essence of the data for discriminative applications. A mixture of coefficients model is proposed to carry out this decomposition, which is termed the Dirty Model:[31]

$$\min_{\mathbf{A},\mathbf{B}} \frac{1}{2}||\mathbf{X} - \mathbf{D}(\mathbf{A} + \mathbf{B})||_F^2 + \lambda_1||\mathbf{A}||_{1,\infty} + \lambda_2||\mathbf{B}||_{1,1}, \tag{4.2.4}$$

where $||\cdot||_F$ denotes the Frobenius norm, $\ell_{1,\infty}$-norm encourages the block sparsity and $\ell_{1,1}$-norm promotes sparsity. The Dirty model addresses the drawback of C-HiLasso because $\mathbf{A}$ points out dictionary atoms that are shared across all tasks (similarity) and

B captures those that are uniquely utilized by individual task (difference). However, it assumes no label differences between dictionary atoms. Thus, it lacks the group information that indicates sub-dictionaries for different classes.

In summary, there are three key factors one could consider when designing DL methods for classification: sparsity, group structure and if possible, within-group similarity. Sparsity makes it easier to interpret the data and brings in the possibility of identifying the difference in a high-dimensional feature space. Group structure naturally coincides with the label information in the classification problem. It enforces the labels implicitly, thus will not increase the size of the problem. Within-group similarity can be used to further refine the group structure by finding a smaller set of dictionary atoms in each group that can resemble all the data in each class.

Inspired by these observations, we first propose Hierarchical Dictionary Learning (HiDL) as in Fig 4.2. Different from sparsity or locality driven DL approaches, HiDL strictly enforces the group boundary between different classes, thus works better when the data is close to the class boundary. We generalize HiDL to the scenario when multiple training data samples are correlated (also called multi-task learning for classification problems[26]). We call this generalization Group Structured Dirty Dictionary Learning (GDDL), which combines the group structure with the Dirty Model and find the shared atoms as well as unique atoms in each class. This could further strengthen the locality within each group since the shared dictionary atoms will be more compact in a small neighborhood as in Fig 4.2(d). Notice that constraint

functions $f_{\mathbf{A}}(\cdot)$ and $f_{\mathbf{D}}(\cdot)$ mentioned in Section 4.1 could also be integrated into HiDL and GDDL. However, we adhere to a simple formulation to better understand the principles that matter in following sections.

4.2.2 Hierarchical Dictionary Learning (HiDL)

When training data has large within-class variability, there exists less similarity among the data from the same class. A properly structured mapping enforced by Hi-Lasso (4.2.3) in DL process can guarantee that dictionary atoms are only updated by training data from the same class. This implicit label consistency between dictionary atoms and data cannot be enforced by either Lasso or LLC. Thus, we propose the Hierarchical Dictionary Learning (HiDL), whose objective function is

$$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{x}_i - \mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1 \sum_{g\in\mathcal{G}} ||\mathbf{a}_{i,[g]}||_2 + \lambda_2||\mathbf{a}_i||_1), \tag{4.2.5}$$

essentially incorporating HiLasso into the DL process. Similar to other methods, HiDL iterates between sparse coding and dictionary update. For sparse coding, we are solving HiLasso problem with a predefined group structure. Optimization based approaches[10,79] or Bayesian approach using structured Spike and Slab prior[60] can be adopted for this purpose.

4.2.2.1 Extending HiDL to Learn from Compressed Data (HiDL-CS)

As pointed out by,[80] sometimes we do not have the luxury to get our hands on data in its original size, but rather in its compressed format (e.g., CS measurements). The high cost of acquiring each signal in its full length motivates us to consider learning a discriminative dictionary from a few CS measurements that not only represents the data well, but also has discriminative power as HiDL. Thus, we propose the Compressed Sensing Hierarchical Dictionary Learning (HiDL-CS), whose objective function is modified to

$$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{y}_i - \mathbf{S}_i\mathbf{D}\mathbf{a}_i||_2^2 + \lambda_1 \sum_{g\in\mathcal{G}} ||\mathbf{a}_{i,[g]}||_2 + \lambda_2||\mathbf{a}_i||_1), \tag{4.2.6}$$

Note that we adopt different sensing matrices $\mathbf{S}_i$ for different training data, so that we would not lose information when projecting the training signals onto one low dimensional subspace.

Dictionary Update: Since the sparse coding step could be simply modified by changing the dictionary $\mathbf{D}$ to $\mathbf{SD}$, we just need to derive a dictionary update scheme for HiDL-CS. Here we rewrite the objective function for dictionary update as

$$\min_{\mathbf{D}} \sum_{i=1}^{N} (\frac{1}{2}||\mathbf{y}_i - \mathbf{S}_i \sum_{k=1}^{K} \mathbf{d}_k\mathbf{a}_{i,k}||_2^2), \tag{4.2.7}$$

where $\mathbf{d}_k$ is the k-th dictionary atom of $\mathbf{D}$ and $\mathbf{a}_{i,k}$ is the k-th value of the sparse coefficient vector $\mathbf{a}_i$. As pointed out in,[80] we could rewrite the objective function to be,

$$\min_{\mathbf{D}} \sum_{i=1}^{N} (\frac{1}{2}||\hat{\mathbf{y}_i^k} - \mathbf{S}_i\mathbf{d}_k\mathbf{a}_{i,k}||_2^2), \tag{4.2.8}$$

where $\hat{\mathbf{y}_i^k} = \mathbf{y}_i - \mathbf{S}_i \sum_{j\neq k} \mathbf{d}_j\mathbf{a}_{i,j}$. Thus, we could set the partial derivative of the new objective function to zero, which makes,

$$\sum_{i=1}^{N} (|a_{i,k}|^2\mathbf{S}_i^\top\mathbf{S}_i\hat{\mathbf{d}}_k - a_{i,k}\mathbf{S}_i^\top\hat{\mathbf{y}_i^k}) = \mathbf{0}, \tag{4.2.9}$$

which yields each dictionary atom (before normalization) as,

$$\hat{\mathbf{d}}_k = (\sum_{i=1}^{N} |a_{i,k}|^2\mathbf{S}_i^\top\mathbf{S}_i)^\dagger \sum_{i=1}^{N} a_{i,k}\mathbf{S}_i\hat{\mathbf{y}_i^k}. \tag{4.2.10}$$

Taking normalization, we could get the learned dictionary atom as,

$$\mathbf{d}_k = \hat{\mathbf{d}}_k/||\hat{\mathbf{d}}_k||_2. \tag{4.2.11}$$

4.2.3 Group Structured Dirty Dictionary Learning (GDDL)

HiDL makes the assumption that different tasks are independent on how they select dictionary atoms, therefore the sparse coding step for each task is carried out separately. In some applications, training data in each class is tightly clustered, indicating a large within-class similarity. For instance, pictures of the same person taken under different illumination conditions in face recognition tasks can still be visually identified to belong to the same class. Such correlation among training data with the same label is not properly captured by HiDL. Therefore, we generalize HiDL for a multi-task problem and get Group Structured Dirty Model Dictionary Learning (GDDL) as below:

$$\min_{\mathbf{D},\mathbf{A},\mathbf{B}} \sum_{c=1}^{C} (\frac{1}{2}||\mathbf{X}_c - \mathbf{D}(\mathbf{A}_c + \mathbf{B}_c)||_F^2 + \lambda_1||\mathbf{A}_c||_{1,2} + \lambda_2||\mathbf{B}_c||_{1,1} + \lambda_3 \sum_{g\in\mathcal{G}} ||\mathbf{A}_{c,[g]}||_F + \lambda_4 \sum_{g\in\mathcal{G}} ||\mathbf{B}_{c,[g]}||_F), \tag{4.2.12}$$

where $\mathbf{X}_c$ is all training data from c-th class, while $\mathbf{A}_c$ and $\mathbf{B}_c$ are the sub-matrices in $\mathbf{A}$ and $\mathbf{B}$ consisting of columns for class c, respectively. Furthermore, $\mathbf{A}_{c,[g]}$ and $\mathbf{B}_{c,[g]}$ are the sub-matrices by extracting rows with indices in group g from $\mathbf{A}_c$ and $\mathbf{B}_c$, respectively. The first three terms impose the Dirty Model with $\ell_{1,2}$-norm and $\ell_{1,1}$-norm for promoting row sparsity and sparsity, respectively. Since the dictionary $\mathbf{D}$

contains sub-dictionaries from all classes, extra constraints are needed to guarantee the active rows from $\mathbf{A}_c$ and active indices from $\mathbf{B}_c$ fall into the same group, respectively. Inspired by C-HiLasso, we use the collaborative Group Lasso regularizers $\sum_{g\in\mathcal{G}} ||\mathbf{A}_{c,[g]}||_F$ and $\sum_{g\in\mathcal{G}} ||\mathbf{B}_{c,[g]}||_F$ to force the group boundary.

Sparse coding by Group Structured Dirty Model: We call the sparse coding step of GDDL $--$ Group Structured Dirty Model (GSDM). It can also be interpreted as a generalization of C-HiLasso and the Dirty Model. When different tasks do not have to share atoms, the sparse coding step of (4.2.12) turns into

$$\min_{\mathbf{B}} \frac{1}{2}||\mathbf{X}_c - \mathbf{D}\mathbf{B}_c||_F^2 + \lambda_2||\mathbf{B}_c||_{1,1} + \lambda_4 \sum_{g\in\mathcal{G}} ||\mathbf{B}_{c,[g]}||_F, \forall\, c, \tag{4.2.13}$$

which is exactly C-HiLasso enforcing both group sparsity and within-group sparsity. When there is no label difference between dictionary atoms (no group structure), the sparse coding step of (4.2.12) becomes

$$\min_{\mathbf{A},\mathbf{B}} \frac{1}{2}||\mathbf{X}_c - \mathbf{D}(\mathbf{A}_c + \mathbf{B}_c)||_F^2 + \lambda_1||\mathbf{A}_c||_{1,2} + \lambda_2||\mathbf{B}_c||_{1,1}, \forall\, c, \tag{4.2.14}$$

which is the Dirty Model with decomposition of row sparsity (replacing $\ell_{1,\infty}$-norm with $\ell_{1,2}$-norm) and sparsity terms.

Nevertheless, there are three key differences between GSDM and the Dirty model. First, GSDM extends the Dirty model by adding another layer of group sparsity, which is illustrated in Fig 4.3. Different from the Dirty Model, GSDM enforces all

the activate supports to stay within the same group corresponding to the desired class. It is the same as why HiDL generates more discriminative dictionary than simply concatenating K-SVD dictionaries for each class. As we will show later using experiments, this group structure constraint encourages the dictionary atoms from different classes to compete for being selected in sparse coding. Therefore, dictionary as well as sparse coefficient become more discriminative. Second, the sparse codes are further decomposed into two parts within the group, one with supports shared across tasks and one with unique supports associated with different tasks. Correspondingly, the shared dictionary atoms captures the similarity among tasks. Finally, the Dirty Model is oriented from a reconstruction perspective while the GSDM brings in the group structure for classification. In short, GSDM could uniquely combine sparsity, group structure and within-group similarity (or locality) in a single formulation.

Optimization Approach for GSDM: The Group Structured Dirty Model problem can be reformulated as follows:

$$\min_{\mathbf{A},\mathbf{B}} ||\mathbf{A}_c||_{1,2} + \lambda_2||\mathbf{B}_c||_{1,1} + \lambda_3 \sum_{g\in\mathcal{G}} ||\mathbf{A}_{c,[g]}||_F + \lambda_4 \sum_{g\in\mathcal{G}} ||\mathbf{B}_{c,[g]}||_F$$
$$s.t.\ \mathbf{X}_c - \mathbf{D}(\mathbf{A}_c + \mathbf{B}_c) = \mathbf{0}, \forall\ c, \tag{4.2.15}$$

with the re-scaled regularization parameters (which will not affect the results). We choose the alternating direction method of multipliers (ADMM) as the optimization approach because of its simplicity, efficiency and robustness.[81,82] By introducing two

auxiliary variables $\mathbf{U} \in \mathbb{R}^{K\times N}$ and $\mathbf{V} \in \mathbb{R}^{K\times N}$, this problem can be reformulated as:

$$\begin{aligned}
&\min_{\mathbf{A},\mathbf{B},\mathbf{U},\mathbf{V}} ||\mathbf{U}_c||_{1,2} + \lambda_2||\mathbf{V}_c||_{1,1} \\
&\quad + \sum_{g\in\mathcal{G}} (\lambda_3||\mathbf{U}_{c,[g]}||_F + \lambda_4||\mathbf{V}_{c,[g]}||_F) \\
&\quad s.t.\ \mathbf{A}_c - \mathbf{U}_c = \mathbf{0},\ \mathbf{B}_c - \mathbf{V}_c = \mathbf{0}, \\
&\quad \mathbf{X}_c - \mathbf{D}(\mathbf{A}_c + \mathbf{B}_c) = \mathbf{0}, \forall\ c.
\end{aligned} \tag{4.2.16}$$

Therefore, the augmented Lagrangian function with respect to $\mathbf{A}$, $\mathbf{B}$, $\mathbf{U}$, and $\mathbf{V}$ can be formed as:

$$\begin{aligned}
L_\mu(\mathbf{A},\mathbf{B},\mathbf{U},\mathbf{V}) = &\sum_{c=1}^{C} \Big(||\mathbf{U}_c||_{1,2} + \lambda_2||\mathbf{V}_c||_{1,1} \\
&+\lambda_3 \sum_{g\in\mathcal{G}} ||\mathbf{U}_{c,[g]}||_F + \lambda_4 \sum_{g\in\mathcal{G}} ||\mathbf{V}_{c,[g]}||_F \Big) \\
&+tr(\hat{\mathbf{Y}}_1, \mathbf{A} - \mathbf{U}) + tr(\hat{\mathbf{Y}}_2, \mathbf{B} - \mathbf{V}) \\
&+tr(\hat{\mathbf{Y}}_3, \mathbf{X} - \mathbf{D}(\mathbf{A} + \mathbf{B})) \\
&+\frac{\mu}{2}\Big(||\mathbf{A} - \mathbf{U}||_F^2 + ||\mathbf{B} - \mathbf{V}||_F^2 \\
&+||\mathbf{X} - \mathbf{D}(\mathbf{A} + \mathbf{B})||_F^2 \Big)
\end{aligned} \tag{4.2.17}$$

where $\hat{\mathbf{Y}}_1$, $\hat{\mathbf{Y}}_2$, $\hat{\mathbf{Y}}_3$ are the Lagrangian multipliers for equality constraints and $\mu > 0$ is a penalty parameter. The augmented Lagrangian function (4.2.17) can be minimized over $\mathbf{A}$, $\mathbf{B}$, $\mathbf{U}$, and $\mathbf{V}$ iteratively by fixing one variable at a time and updating the

others. The entire algorithm is summarized in Algorithm 3, where we let $\mathbf{Y}_1 = \frac{\hat{\mathbf{Y}}_1}{\mu}$, $\mathbf{Y}_2 = \frac{\hat{\mathbf{Y}}_2}{\mu}$, $\mathbf{Y}_3 = \frac{\hat{\mathbf{Y}}_3}{\mu}$. And $\mathbf{Y}_{1,c}$, $\mathbf{Y}_{2,c}$ and $\mathbf{Y}_{3,c}$ are the submatrices with columns corresponding to c-th class in $\mathbf{Y}_1$, $\mathbf{Y}_2$ and $\mathbf{Y}_3$, respectively.

Deriving the Proximal Operators for GSDM: The key steps in Algorithm 3 are Step 4 and 6. Because GSDM could be regarded as an generalization of C-HiLasso as pointed out by (4.2.15), Step 6 can also be solved using the same operator as for C-HiLasso ((III.14),[10]), which is derived using proximal methods.[83,84] The key step of using proximal methods to solve GSDM is the computation of the proximal operator, that can be solved exactly with a complexity linear, or close to linear, in the number of dictionary elements.

First, we will introduce the proximal operators that encourage row sparsity, component-wise sparsity and group sparsity.[85] Inside each group, the proximal operator $\text{Prox}_{\kappa_2,\Omega_{1,2}}$ that encourages row sparsity is:

$$\text{Prox}_{\kappa_2,\Omega_{1,2}}(\mathbf{v}_{(j,:)}) = \left(1 - \frac{\kappa_2}{||\mathbf{v}_{(j,:)}||_2}\right)_+ \mathbf{v}_{(j,:)} \tag{4.2.18}$$

where $\mathbf{v}_{(j,:)}$ is defined as j-th row of $\mathbf{V}$ and $(x)_+ := \max(x, 0)$. So it will zero out rows with l_2-norms below the threshold κ_2. The proximal operator $\text{Prox}_{\kappa_4,\Omega_{1,1}}$ for component-wise sparsity is:

$$\text{Prox}_{\kappa_4,\Omega_{1,1}}(v_{j,i}) = \left(1 - \frac{\kappa_4}{|v_{j,i}|}\right)_+ v_{j,i} \tag{4.2.19}$$

Algorithm 3: Solving Group Structured Dirty Model Problem with ADMM

Input: Training data $\mathbf{X}$, learned dictionary $\mathbf{D}$, group structure $\mathcal{G}$, scalar $\rho = 1.1$, and regularization parameters λ_2,λ_3,λ_4;

Output: Sparse codes $\mathbf{A}$ and $\mathbf{B}$;

Initializing $\mathbf{A}^0 = \mathbf{0}$, $\mathbf{B}^0 = \mathbf{0}$, $\mathbf{Y}_1^0 = \mathbf{0}$, $\mathbf{Y}_2^0 = \mathbf{0}$, $\mathbf{Y}_3^0 = \mathbf{0}$, $\mu = 1$, $\mu_{max} = 10^6$, $k = 0$;

for $c = 1, ..., C$ **do**

  while *not converged* **do**

    Fix $\mathbf{A}_c$, $\mathbf{B}_c$, $\mathbf{V}_c$ and update $\mathbf{U}_c$ by:

$$\begin{aligned}\mathbf{U}_c^{k+1} &= \arg\min L_\mu(\mathbf{A}_c^k, \mathbf{B}_c^k, \mathbf{U}_c, \mathbf{V}_c^k) \\ &= \mathrm{Prox}_{\Omega_{\mathcal{G},(1,2)}}(\mathbf{A}_c^k + \mathbf{Y}_{1,c}^k)\end{aligned}$$

    Fix $\mathbf{B}_c$, $\mathbf{U}_c$, $\mathbf{V}_c$ and update $\mathbf{A}_c$ by:

$$\begin{aligned}\mathbf{A}_c^{k+1} &= \arg\min L_\mu(\mathbf{A}_c, \mathbf{B}_c^k, \mathbf{U}_c^{k+1}, \mathbf{V}_c^k) \\ &= (\mathbf{D}^\top\mathbf{D} + \mathbf{I})^{-1} \\ &\quad [\mathbf{D}^\top(\mathbf{X}_c + \mathbf{Y}_{3,c}^k - \mathbf{D}\mathbf{B}_c^k) + \mathbf{U}_c^{k+1} - \mathbf{Y}_{1,c}^k]\end{aligned}$$

    Fix $\mathbf{A}_c$, $\mathbf{B}_c$, $\mathbf{U}_c$ and update $\mathbf{V}_c$ by:

$$\begin{aligned}\mathbf{V}_c^{k+1} &= \arg\min L_\mu(\mathbf{A}_c^{k+1}, \mathbf{B}_c^k, \mathbf{U}_c^{k+1}, \mathbf{V}_c) \\ &= \mathrm{Prox}_{\Omega_{\mathcal{G},(1,1)}}(\mathbf{B}_c^k + \mathbf{Y}_{2,c}^k)\end{aligned}$$

    Fix $\mathbf{A}_c$, $\mathbf{U}_c$, $\mathbf{V}_c$ and update $\mathbf{B}_c$ by:

$$\begin{aligned}\mathbf{B}_c^{k+1} &= \arg\min L_\mu(\mathbf{A}_c^{k+1}, \mathbf{B}_c, \mathbf{U}_c^{k+1}, \mathbf{V}_c^{k+1}) \\ &= (\mathbf{D}^\top\mathbf{D} + \mathbf{I})^{-1} \\ &\quad [\mathbf{D}^\top(\mathbf{X}_c + \mathbf{Y}_{3,c}^k - \mathbf{D}\mathbf{A}_c^{k+1}) + \mathbf{V}_c^{k+1} - \mathbf{Y}_{2,c}^k]\end{aligned}$$

    Update Lagrange multipliers $\mathbf{Y}_{1,c}$, $\mathbf{Y}_{2,c}$, $\mathbf{Y}_{3,c}$:

$$\begin{aligned}\mathbf{Y}_{1,c}^{k+1} &= \mathbf{Y}_{1,c}^k + \mathbf{A}_c^{k+1} - \mathbf{U}_c^{k+1} \\ \mathbf{Y}_{2,c}^{k+1} &= \mathbf{Y}_{2,c}^k + \mathbf{B}_c^{k+1} - \mathbf{V}_c^{k+1} \\ \mathbf{Y}_{3,c}^{k+1} &= \mathbf{Y}_{3,c}^k + \mathbf{X}_c - \mathbf{D}(\mathbf{A}_c^{k+1} + \mathbf{B}_c^{k+1})\end{aligned}$$

    Update penalty parameter $\mu = \min(\mu_{max}, \rho\mu)$

    Increment k.

return *Estimated sparse codes* $\mathbf{A}$ *and* $\mathbf{B}$.

where $v_{j,i}$ is the value of $\mathbf{V}$ at the coordinate $[j, i]$. Finally, the proximal operator for group sparsity is:

$$\text{Prox}_{\kappa_1,\Omega_\mathcal{G}}(\mathbf{V}_{[g]}) = \left(1 - \frac{\kappa_1}{||\mathbf{V}_{[g]}||_F}\right)_+ \mathbf{V}_{[g]} \tag{4.2.20}$$

where $\mathbf{V}_{[g]}$ is the sub-matrix with rows indexed by group g. It has the effect of zeroing or keeping coefficients in the same group all together.

In our case, the GSDM contains group sparsity structure and row sparsity structure for $\mathbf{A}_c$ and it contains group sparsity structure and element-wise sparsity structure for $\mathbf{B}_c$. Both can be interpreted as composite norm in a hierarchical sparse coding procedure. As pointed out in,[86] the proximal operators associated with the composite norm in hierarchical sparse coding can be obtained by the composition of the proximal operators as long as the sparsity structures follow the right order. This order is termed as a total order relationship or tree-structured sets of groups (Definition 1,[22]), which requires that the two groups are either disjoint or one is included in the other. Both cases in GSDM satisfy the total order relationship because either the individual index or the individual row is included in groups as clearly shown in Fig 4.3(b). After establishing the total order relationship, the proximal operators for composite norm could be constructed by applying the proximal operators for smaller groups first, followed by the ones for larger groups. Therefore, the corresponding operators for Step 4 and 6 in Algorithm 3 can be derived as below:

$$\text{Prox}_{\Omega_{\mathcal{G},(1,2)}} = \text{Prox}_{\kappa_1,\Omega_\mathcal{G}} \circ \text{Prox}_{\kappa_2,\Omega_{1,2}} \tag{4.2.21}$$

and

$$\text{Prox}_{\Omega_{\mathcal{G},(1,1)}} = \text{Prox}_{\kappa_3,\Omega_{\mathcal{G}}} \circ \text{Prox}_{\kappa_4,\Omega_{1,1}} \tag{4.2.22}$$

where $\text{Prox}_{\kappa_1,\Omega_{\mathcal{G}}}$ and $\text{Prox}_{\kappa_3,\Omega_{\mathcal{G}}}$ are the proximal operators for group sparsity, whereas $\text{Prox}_{\kappa_2,\Omega_{1,2}}$ and $\text{Prox}_{\kappa_4,\Omega_{1,1}}$ promotes the selection of only a few non-zero rows and elements, respectively. So $\text{Prox}_{\Omega_{\mathcal{G},(1,2)}}$ for Step 4 can be readily computed by applying first the proximal operator associated with the $\ell_{1,2}$-norm (row-wise soft-thresholding) and then the one associated with group sparsity $\text{Prox}_{\kappa_1,\Omega_{\mathcal{G}}}$. Similarly, the C-HiLasso operator $\text{Prox}_{\Omega_{\mathcal{G},(1,1)}}$ for Step 6 is just applying the element-wise soft-thresholding and then the group thresholding, which is same as in.[10] Here, we have $\kappa_1 = \frac{\lambda_3}{\mu}, \kappa_2 = \frac{1}{\mu}, \kappa_3 = \frac{\lambda_4}{\mu}, \kappa_4 = \frac{\lambda_2}{\mu}$. Note that GSDM separates the sparse code into shared indices $\mathbf{A}_c$ and unique indices $\mathbf{B}_c$, thus the selected group in $\mathbf{A}_c$ could be different from that in $\mathbf{B}_c$. To avoid such situation, we enforce the same group selection by always using the group selected by row-sparsity term, because it is a stronger constraint than sparsity.

Dictionary Update: We adopt the method of block coordinate descent with a warm start to update one dictionary atom at a time.[19] Other methods such as Singular Value Decomposition (SVD)[16] or Method of Optimal Directions (MOD)[65] could also be used and yield similar results. Meanwhile, we choose block coordinate descent method to show in Section III that under certain conditions this approach forces the dictionary atoms to be updated in the same subspace. Using the facts that $||\mathbf{X} - \mathbf{DA}||_F^2 = tr[(\mathbf{X} - \mathbf{DA})(\mathbf{X} - \mathbf{DA})^\top]$ and trace is invariant under cyclic

permutations, the objective function of the dictionary update step can be changed to:

$$\min_{\mathbf{D}} \frac{1}{2} tr(\mathbf{D}^\top \mathbf{D}\boldsymbol{\Psi}) - tr(\mathbf{D}^\top \boldsymbol{\Phi}) \tag{4.2.23}$$

where

$$\boldsymbol{\Psi} = \sum_{i=1}^{N} \mathbf{a}_i \mathbf{a}_i^\top, \ \boldsymbol{\Phi} = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{a}_i^\top.$$

Taking the derivative and set it to zero, we obtain the dictionary update procedure as follow:

$$\hat{\mathbf{d}} \leftarrow \frac{1}{\Psi_{j,j}}(\boldsymbol{\phi}_j - \mathbf{D}\boldsymbol{\psi}_j) + \mathbf{d}_j^t, \ \mathbf{d}_j^{t+1} \leftarrow \frac{1}{max(||\hat{\mathbf{d}}||_2, 1)}\hat{\mathbf{d}}, \tag{4.2.24}$$

where $\Psi_{j,j}$ is the value of $\boldsymbol{\Psi}$ at coordinate $[j, j]$ with $\mathbf{d}_j^t$ and $\mathbf{d}_j^{t+1}$ being the j-th atom at t-th and $t+1$-th iterations, respectively. The last step in (4.2.24) guarantees all dictionary atoms to have unit norms. The dictionary is initialized with random sampling of training data and the motivation will be explained in Section 4.3 from a theoretical standpoint.

Putting together the sparse coding and dictionary update processes, we complete the algorithm for GDDL as presented in Algorithm 4.

4.2.4 Classification approach

For classification, we choose a linear classifier for its simplicity and the purpose of fair comparison with results of other techniques, although advanced classification

Algorithm 4: Group Structured Dirty Dictionary Learning (GDDL)

Input: Labeled training data $\mathbf{x}_i, i = 1, .., N$, the group structure $\mathcal{G}$, scalar $\rho = 1.1$, and regularization parameters λ_1 and λ_2;

Output: Dictionary $\mathbf{D}$ and sparse code $\mathbf{A}$ and $\mathbf{B}$;

Initializing $\mathbf{D}^0$ by random sampling from training data of each class and $t = 0$;

while *not converged* **do**

  Fix $\mathbf{D}^t$ and update $\mathbf{A}^{t+1}$ and $\mathbf{B}^{t+1}$ using Algorithm 1. (For HiDL, use convex optimization to solve HiLasso[10] instead.)

  Fix $\mathbf{A}^{t+1}$ as well as $\mathbf{B}^{t+1}$ and update $\mathbf{D}^{t+1}$ using (4.2.22) (or (4.2.8) for learning from CS measurements).

  Increment t.

return *dictionary* $\mathbf{D}$ *and sparse code* $\mathbf{A}$ *and* $\mathbf{B}$.

techniques (i.e., SRC[48]) could potentially lead to better performances. The linear classifier $\mathbf{W} \in \mathbb{R}^{C \times K}$ is found by:

$$\mathbf{W}^\top = (\mathbf{A}\mathbf{A}^\top + \eta \mathbf{I})^{-1} \mathbf{A}\mathbf{L}^\top \tag{4.2.25}$$

where $\mathbf{A}$ is the learned sparse codes for training data from either HiDL or GDDL. The matrix $\mathbf{L} \in \mathbb{R}^{C \times N}$ provides the label information for training data. If training data $\mathbf{x}_i$ belongs to the c-th class, then $L_{c,i}$ is one and all other elements in the same column are zero. The parameter η controls the trade-off between the classification accuracy and the smoothness of the classifier. Because we enforce group structure in HiDL and GDDL, the sparse coefficient of training data $\mathbf{A}$ has a block diagonal structure. It also makes our linear classifier $\mathbf{W}$ form a block diagonal structure. Therefore, the non-zero sparse coefficients on undesired support of test data could be zeroed out by this block diagonal classifier. We will further explore the condition for $\mathbf{A}$ to have the

block diagonal structure in Section 4.3. For each test data $\mathbf{x}$, we find its sparse code by solving HiLasso with the learned dictionary $\mathbf{D}$, then apply the classifier $\mathbf{W}$ to get the label vector $\mathbf{l}_{est}$. The test data is then assigned to the class $c = \arg\max_c \mathbf{l}_{est}$.

For GDDL, we only use the shared sparse coefficient $\mathbf{A}$ to train the classifier. This has the benefit of making the sparse coefficients more discriminative because they are mapped to the dictionary atoms that are around the center of the cluster. Therefore we could increase the between-class distance among the sparse codes of different classes. For the subsequent classification step, we only feed the shared sparse code $\mathbf{a}$ into the classifier.

4.3 Theoretical Analysis

In this section, we will use HiDL to investigate its theoretical guarantees, then justify the benefit and tradeoff of using structured sparsity in DL for classification. Currently, most of the theoretical analysis of DL focused on the properties of the learned dictionary from a reconstruction perspective. It has been shown that given enough noiseless or small Gaussian noise contaminated training data, using ℓ_1-or ℓ_0-norm regularization in DL leads to a dictionary $\mathbf{D}$, which is a local minimum around the groundtruth with high probability.[87–89] However, little theoretical effort is focused on analyzing the discrimination power of the learned dictionary, which we will explore in this section.

The DL problem is non-convex, making the direct analysis of its solution not trivial. Inspired by the connection between K-SVD and K-means, we interpret the sparse coding stage as analogous to sparse subspace clustering (SSC),[64] and the dictionary learning step is essentially a way of learning the basis for different subspaces. However, there are two key differences between HiDL and SSC.

(i) HiDL is proposed for supervised learning and SSC is developed for unsupervised learning, thus the first difference is the availability of the group structure (label) information. In HiDL, different groups correspond to different subspaces (labels). This in turn leads to the enforcement of group structure sparsity rather than ℓ_1-norm, which is later shown to make the condition for perfect sparse decomposition stricter. However, this price is paid to make the sparse code more discriminative by guaranteeing perfect block structure to separate different classes;

(ii) To represent the subspaces, HiDL uses learned dictionary atoms while SSC uses data directly. Therefore, the success of SSC only depends on the success recovery of sparse coding step since subspace representation (data) is fixed. While for HiDL, dictionary atoms are updated in every iteration so we also need to demonstrate that the dictionary update will not jeopardize the representation of the subspaces. This motivates us to take an inductive approach for analysis.

In this section, we assume that the sparse decomposition is exact so all training data have a perfect decomposition $\mathbf{x}_i = \mathbf{D}\mathbf{a}_i$. Scalings of λ_1 and λ_2 do not affect the optimal solution, so we replace them by a single parameter λ. Now the sparse coding

step of HiDL could be re-written as:

$$\min_{\mathbf{A}} \lambda \sum_{g \in \mathcal{G}} ||\mathbf{a}_{i,[g]}||_2 + (1-\lambda)||\mathbf{a}_i||_1 \; s.t. \; \mathbf{x}_i = \mathbf{D}\mathbf{a}_i \; , \forall i \tag{4.3.1}$$

Then, we borrow the concepts of independent and disjoint subspaces from SSC framework[64] as below.

Definition 1: *Given a collection of subspaces* $\{\mathcal{S}_c\}_{c=1}^{C}$. *If* $dim(\oplus_{c=1}^{C}\mathcal{S}_c) = \sum_{c=1}^{C} dim(\mathcal{S}_c)$, *then* $\{\mathcal{S}_c\}_{c=1}^{C}$ *is independent where* $\oplus$ *denotes the direct sum operator. If every pair of subspaces intersect only at the origin, then* $\{\mathcal{S}_c\}_{c=1}^{C}$ *is disjoint.*

The index of subspaces ($c = 1, ..., C$) is purposely chosen to be same as the class labels to emphasize the correspondence between sub-dictionary $\mathbf{D}_c$ and subspace $\mathcal{S}_c$ (class label). To characterize two disjoint subspaces,[64] also defined an important notion: the smallest principal angle.

Definition 2: *The smallest principle angle* θ_{c_1,c_2} *between two disjoint subspaces* $\mathcal{S}_{c_1}$ *and* $\mathcal{S}_{c_2}$ *is:*

$$cos(\theta_{c_1,c_2}) = \max_{\mathbf{v}_{c_1} \in \mathcal{S}_{c_1}, \mathbf{v}_{c_2} \in \mathcal{S}_{c_2}} \frac{\mathbf{v}_{c_1}^{\top}\mathbf{v}_{c_2}}{||\mathbf{v}_{c_1}||_2||\mathbf{v}_{c_2}||_2}$$

which gives $cos(\theta_{c_1,c_2}) \in [0, 1)$.

4.3.1 Performance Analysis

With the aforementioned notations, we use an induction approach to show the following results.

Theorem 1: *Given enough noiseless training data points spanning all C subspaces $\{\mathcal{S}_c\}_{c=1}^{C}$ of dimension $\{r_c\}_{c=1}^{C}$. If we train the dictionary using HiDL, and both Lemma 1 and Lemma 2 are satisfied, the noiseless test data from the same C subspaces will have a perfect block sparse representation with respect to the trained dictionary.*

To be more specific, we will show two properties for the sparse coding and dictionary update stages hold under certain conditions.

(i) Support recovery property: starting in the sparse coding stage, the sparse code $\mathbf{a}$ for training data $\mathbf{x}$ of c-th class will have a perfect block structure such that $\mathbf{a}_c \neq 0$ and $\mathbf{a}_{-c} = 0$, where $\mathbf{a}_c$ and $\mathbf{a}_{-c}$ indicate the sub-vectors corresponding to the subspace $\mathcal{S}_c$ and all other subspaces except $\mathcal{S}_c$;

(ii) Subspace consistency property: then in the dictionary learning stage, the dictionary update procedure (4.2.24) guarantees the dictionary atoms to be updated in the same subspace.

Support recovery property: Similar to Theorem 1 in,[64] it is straightforward to see the support recovery property holds for the case of independent subspace. We can show that the support recovery property also holds for the disjoint subspace case as long as the following Lemma 1 holds.

Lemma 1: *(Disjoint Subspace Case) Consider a collection of data points drawn from C disjoint subspaces $\{\mathcal{S}_c\}_{c=1}^{C}$ of dimension $\{r_c\}_{c=1}^{C}$. If the condition*

$$\sigma_{min}(\mathbf{D}_{c_1}) > \frac{\left(\lambda + (1-\lambda)\sqrt{K_{c_1}}\right) \max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})}{\frac{\lambda}{\sqrt{K_{-c_1}}} + (1-\lambda)} \tag{4.3.2}$$

is satisfied, then for every nonzero input $\mathbf{x} \in \mathcal{S}_c$*, (4.3.1) recovers a perfect subspace sparse structure, i.e.,* $\mathbf{a}_c \neq 0$ *and* $\mathbf{a}_{-c} = 0$*. Note that* K_{c_1} *and* K_{-c_1} *are the size of* $\mathbf{D}_{c_1}$ *and* $\mathbf{D}_{-c_1}$*, respectively. And* $\sigma_{min}(\mathbf{D}_{c_1})$ *is the smallest singular value of* $\mathbf{D}_{c_1}$*.*

From Lemma 1, we can see that the condition for a perfect block structure recovery is more likely to be satisfied if the dictionary size is smaller, the smallest principle angle between two disjoint subspace is larger or the smallest singular value of $\mathbf{D}_{c_1}$ is larger.

Subspace consistency property: We then show that dictionary update procedure (4.2.24) gives us Lemma 2 as below.

Lemma 2: *Suppose the training data* $\mathbf{x}$ *belongs to the c-th class. Assume that each sub-dictionary is full-rank. At the t-th iteration, if the dictionary atom* $\mathbf{d}_j^{t-1} \in \mathcal{S}_c$ *and sparse coefficient* $\mathbf{a}^t$ *from the previous sparse coding stage has a block structure such that* $\mathbf{a}_c^t \neq 0$ *and* $\mathbf{a}_{-c}^t = 0$*, then the updated dictionary atom* $\mathbf{d}_j^t \in \mathcal{S}_c$*.*

This indicates that if the sparse coding step generates a sparse coefficient $\mathbf{a}$ with a perfect block structure, then the dictionary update will guarantee the dictionary atoms to be updated in the same subspace. This in turn will reinforce the disjoint subspace condition required for sparse coding step of next iteration (Lemma 1).

4.3.2 Proof Proof for Support Recovery Property

Similar to Theorem 1 in,[64] it is straightforward to see the support recovery property holds for the case of independent subspace as in following Lemma.

Lemma: *(Independent Subspace Case) Suppose the data are drawn from C subspaces* $\{\mathcal{S}_c\}_{c=1}^{C}$ *of dimension* $\{r_c\}_{c=1}^{C}$. *Let* $\mathbf{D}_c$ *denotes the sub-dictionary for subspace* $\mathcal{S}_c$ *and* $\mathbf{D}_{-c}$ *denotes the sub-dictionary for all other subspaces except* $\mathcal{S}_c$. *Assume that every sub-dictionary* $\mathbf{D}_c$ *is full column rank. If these subspaces are independent, then for every input* $\mathbf{x} \in \mathcal{S}_c$, *(4.3.1) recovers a perfect subspace-sparse structure, i.e., the resulting solutions have* $\mathbf{a}_c^* \neq 0$ *and* $\mathbf{a}_{-c}^* = 0$.

For the disjoint subspace case, we define $\mathbf{z}_{c_1}$ and $\mathbf{z}_{-c_1}$ as below:

$$\mathbf{z}_{c_1} = \arg\min \lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{[g]}||_2 + (1-\lambda)||\mathbf{z}||_1 \; s.t. \; \mathbf{x} = \mathbf{D}_{c_1}\mathbf{z}$$

and

$$\mathbf{z}_{-c_1} = \arg\min \lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{[g]}||_2 + (1-\lambda)||\mathbf{z}||_1 \; s.t. \; \mathbf{x} = \mathbf{D}_{-c_1}\mathbf{z}.$$

Then, we could derive following Lemma for the disjoint subspace case.

Lemma: *(Disjoint Subspace Case) Given the same data and dictionary as in the independent subspace case above. If these subspaces are disjoint, then (4.3.1) recovers a perfect subspace sparse structure if and only if for all nonzero* $\mathbf{x} \in \mathcal{S}_{c_1} \cap \oplus_{c_2 \neq c_1} \mathcal{S}_{c_2}$,

$$\lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{c_1,[g]}||_2 + (1-\lambda)||\mathbf{z}_{c_1}||_1 < \lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{-c_1,[g]}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1.$$

Note that $\mathbf{z}_{c_1,[g]}$ *and* $\mathbf{z}_{-c_1,[g]}$ *are the sub-vectors of* $\mathbf{z}_{c_1}$ *and* $\mathbf{z}_{-c_1}$ *defined by group* g.

The condition for the disjoint subspace case in previous Lemma is related to the sparse coefficients of the data, which is largely dependent on the algorithm. To be more intuitive, it is thus better to directly relate these requirements to either the dictionary or the data, which gives the following Lemma 1.

Lemma 1: *(Disjoint Subspace Case) Consider a collection of data points drawn from C disjoint subspaces $\{\mathcal{S}_c\}_{c=1}^{C}$ of dimension $\{r_c\}_{c=1}^{C}$. If the condition*

$$\sigma_{min}(\mathbf{D}_{c_1}) > \frac{\left(\lambda + (1-\lambda)\sqrt{K_{c_1}}\right) \max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})}{\frac{\lambda}{\sqrt{K_{-c_1}}} + (1-\lambda)} \tag{4.3.3}$$

is satisfied, then for every nonzero input $\mathbf{x} \in \mathcal{S}_c$, (4.3.1) recovers a perfect subspace sparse structure, i.e., $\mathbf{a}_c \neq 0$ and $\mathbf{a}_{-c} = 0$. Note that K_{c_1} and K_{-c_1} are the size of $\mathbf{D}_{c_1}$ and $\mathbf{D}_{-c_1}$, respectively. And $\sigma_{min}(\mathbf{D}_{c_1})$ is the smallest singular value of $\mathbf{D}_{c_1}$.

Proof:

Step 1: First, we will find the upper bound β_{c_1} for the left side of the original condition in Lemma 2, $\lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{c_1,[g]}||_2 + (1-\lambda)||\mathbf{z}_{c_1}||_1$. Since data $\mathbf{x} \in \mathcal{S}_{c_1} \cap \oplus_{c_2 \neq c_1} \mathcal{S}_{c_2}$ and $\mathbf{D}_{c_1}$ is full column rank, we have,

$$\mathbf{x} = \mathbf{D}_{c_1}\mathbf{z}_{c_1} \Rightarrow \mathbf{z}_{c_1} = (\mathbf{D}_{c_1}^{\top}\mathbf{D}_{c_1})^{-1}\mathbf{D}_{c_1}^{\top}\mathbf{x} \tag{4.3.4}$$

Since the subspace structure matches the group structure, we have

$$\lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{c_1,[g]}||_2 + (1-\lambda)||\mathbf{z}_{c_1}||_1 = \lambda||\mathbf{z}_{c_1}||_2 + (1-\lambda)||\mathbf{z}_{c_1}||_1.$$

Applying the vector norm property yields

$$\lambda||\mathbf{z}_{c_1}||_2 + (1-\lambda)||\mathbf{z}_{c_1}||_1 \leq \lambda||\mathbf{z}_{c_1}||_2 + (1-\lambda)\sqrt{K_{c_1}}||\mathbf{z}_{c_1}||_2$$

where K_{c_1} is the size of sub-dictionary $\mathbf{D}_{c_1}$. Next, applying (4.3.4) and the matrix norm properties ($||\mathbf{Ax}||_2 \leq ||\mathbf{A}||_{2,2}||\mathbf{x}||_2$ and $||\mathbf{A}^{-1}||_{2,2} = \frac{1}{\sigma_{min}(\mathbf{A})}$) , we have

$$\left(\lambda + (1-\lambda)\sqrt{K_{c_1}}\right)||\mathbf{z}_c||_2 = \left(\lambda + (1-\lambda)\sqrt{K_{c_1}}\right)||(\mathbf{D}_{c_1}^\top \mathbf{D}_{c_1})^{-1}\mathbf{D}_{c_1}^\top \mathbf{x}||_2$$
$$\leq \left(\lambda + (1-\lambda)\sqrt{K_{c_1}}\right)||(\mathbf{D}_{c_1}^\top \mathbf{D}_{c_1})^{-1}\mathbf{D}_{c_1}^\top||_{2,2}||\mathbf{x}||_2 = \frac{\lambda + (1-\lambda)\sqrt{K_{c_1}}}{\sigma_{min}(\mathbf{D}_{c_1})}||\mathbf{x}||_2 = \beta_{c_1}$$

Thus, we have derived the upper bound β_{c_1} for the left side of the condition.

Step 2: We will now show the lower bound β_{-c_1} for the right side of the condition $\lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{-c_1,[g]}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1$. Notice that we have

$$\lambda \sum_{g \in \mathcal{G}} ||\mathbf{z}_{-c_1,[g]}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1 = \lambda \sum_{c_2 \in \mathcal{G} \backslash c_1} ||\mathbf{z}_{c_2}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1$$

where we have abused the notation $c_2 \in \mathcal{G}\backslash c_1$ to mean all the groups excluding the

one corresponding to the class c_1. Because

$$\lambda \sum_{c_2 \in \mathcal{G} \setminus c_1} ||\mathbf{z}_{c_2}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1 \geq \lambda ||\mathbf{z}_{-c_1}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1,$$

we can instead find the lower bound for the simplified condition $\lambda||\mathbf{z}_{-c_1}||_2 + (1-\lambda)||\mathbf{z}_{-c_1}||_1$. Based on the definition of $\mathbf{z}_{-c_1}$, we have

$$||\mathbf{x}||_2^2 = \mathbf{x}^\top \mathbf{x} = \mathbf{x}^\top \mathbf{D}_{-c_1} \mathbf{z}_{-c_1}.$$

Using the Holder's inequalities ($|\mathbf{u}^\top \mathbf{v}| \leq ||\mathbf{u}||_\infty ||\mathbf{v}||_1$ and $|\mathbf{u}^\top \mathbf{v}| \leq ||\mathbf{u}||_2 ||\mathbf{v}||_2$) , we obtain

$$||\mathbf{x}||_2^2 = \mathbf{x}^\top \mathbf{D}_{-c_1} \mathbf{z}_{-c_1} \leq ||\mathbf{D}_{-c_1}^\top \mathbf{x}||_\infty ||\mathbf{z}_{-c_1}||_1$$

and

$$||\mathbf{x}||_2^2 = \mathbf{x}^\top \mathbf{D}_{-c_1} \mathbf{z}_{-c_1} \leq ||\mathbf{D}_{-c_1}^\top \mathbf{x}||_2 ||\mathbf{z}_{-c_1}||_2.$$

With the definition of smallest principle angle and the vector norm inequality, we can write

$$||\mathbf{x}||_2^2 \leq \max_{c_2 \neq c_1} cos(\theta_{c_1,c_2}) ||\mathbf{D}_{-c_1}||_{max,2} ||\mathbf{x}||_2 ||\mathbf{z}_{-c_1}||_1$$

and

$$||\mathbf{x}||_2^2 \leq \sqrt{K_{-c_1}} \max_{c_2 \neq c_1} cos(\theta_{c_1,c_2}) ||\mathbf{D}_{-c_1}||_{max,2} ||\mathbf{x}||_2 ||\mathbf{z}_{-c_1}||_2$$

where we use $||\mathbf{D}_{-c_1}||_{max,2}$ to denote the largest ℓ_2-norm of the columns of $\mathbf{D}_{-c_1}$,

which is 1 because we restrict the dictionary atoms in a convex set $\mathcal{D}$ to have unit norm. Therefore, the lower bound for the right side can be shown to be

$$\beta_{-c_1} = \frac{\lambda||\mathbf{x}||_2}{\sqrt{K_{-c_1}}\max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})} + \frac{(1-\lambda)||\mathbf{x}||_2}{\max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})}.$$

Step 3: Combining the lower bound in Step 2 together with the upper bound found in Step 1 gives

$$\frac{\lambda + (1-\lambda)\sqrt{K_{c_1}}}{\sigma_{min}(\mathbf{D}_{c_1})}||\mathbf{x}||_2 < \frac{\lambda||\mathbf{x}||_2}{\sqrt{K_{-c_1}}\max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})} + \frac{(1-\lambda)||\mathbf{x}||_2}{\max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})}$$

which can be simplified to,

$$\sigma_{min}(\mathbf{D}_{c_1}) > \frac{\left(\lambda + (1-\lambda)\sqrt{K_{c_1}}\right)\max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})}{\frac{\lambda}{\sqrt{K_{-c_1}}} + (1-\lambda)}. \quad \square$$

4.3.3 Proof for Subspace Consistency Property

Lemma 2: *Suppose the training data* $\mathbf{x}$ *belongs to the c-th class. Assume that each sub-dictionary is full-rank. At the t-th iteration, if the dictionary atom* $\mathbf{d}_j^{t-1} \in \mathcal{S}_c$ *and sparse coefficient* $\mathbf{a}^t$ *from the previous sparse coding stage has a block structure such that* $\mathbf{a}_c^t \neq 0$ *and* $\mathbf{a}_{-c}^t = 0$*, then the updated dictionary atom* $\mathbf{d}_j^t \in \mathcal{S}_c$*.*

Proof:

Based on the properties of subspace, it suffices to show instead that $\boldsymbol{\phi}_j - \mathbf{D}\boldsymbol{\psi}_j \in \mathcal{S}_c$.

Notice that if $\mathbf{a}_c^t \neq 0$ and $\mathbf{a}_{-c}^t = 0$, then $\mathbf{\Psi}$ will be block diagonal with block structures matching the subspace alignments. Therefore, $\boldsymbol{\psi}_c \neq 0$ and $\boldsymbol{\psi}_{-c} = 0$, i.e. $\mathbf{D}\boldsymbol{\psi}_j \in \mathcal{S}_c$.

Also notice that $\mathbf{\Phi} = \sum_{i=1}^N \mathbf{x}_i \mathbf{a}_i^{t\top} = \mathbf{D}(\sum_{i=1}^N \mathbf{a}_i^* \mathbf{a}_i^{t\top})$, where $\mathbf{a}_i^*$ represents the true sparse code for $\mathbf{x}_i$ that has same block structure. Therefore, $\sum_{i=1}^N \mathbf{a}_i^* \mathbf{a}_i^{t\top}$ has the same block diagonal structure matching the subspace alignments, i.e. $\boldsymbol{\phi}_j \in \mathcal{S}_c$. Therefore, $\mathbf{d}_j^t \in \mathcal{S}_c$. □

4.3.4 Remark

When $\lambda = 0$, the condition (4.3.2) becomes

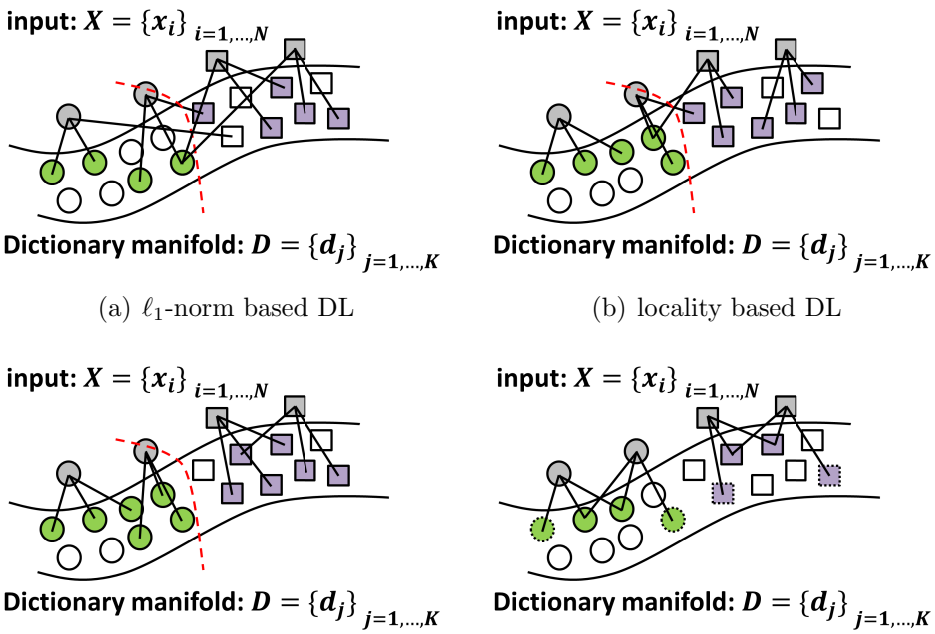
$$\sigma_{min}(\mathbf{D}_{c_1}) > \sqrt{K_{c_1}} \max_{c_1 \neq c_2} cos(\theta_{c_1,c_2})$$

which is exactly the condition derived in Theorem 3 of[64] with the given dictionary having unit norm columns. Moreover, because K_{c_1} is almost always smaller than K_{-c_1}, the condition for HiDL is stricter, which means that the requirement for using structured sparsity is stricter than using ℓ_1-norm. This is the tradeoff paid to recover the sparse code with the right block structure in contrast to no constraints whatsoever on the support by ℓ_1-norm. However, this also gives the benefit of the group structure, which is especially helpful for classification as illustrated in Fig 4.2. Taking a closer look at the condition in (4.3.2), on the left side, the smallest non-zero singular value of the dictionary is bounded from below, yielding a similar effect as the restricted

isometry property (RIP),[5] forcing the transformation between signal domain and coefficient domain to preserve the distance.

The condition in (4.3.2) relates to the size of the dictionary such that the smaller the dictionary size (or indirectly the subspace dimension because the sub-dictionary is full rank), the more likely the condition can be satisfied. When the intrinsic dimension of the signal or the dictionary size is small, it is favorable to choose HiDL. Compared to ℓ_0-norm or ℓ_1-norm regularized DL, HiDL is more likely to recover the perfect block structure, thus could lead to better classification performance. It also suggests that there is a trade-off between using a large dictionary to encode general data distributions versus a small dictionary to ensure its discriminative power.

In short, HiDL has been theoretically shown to be more favorable than the ℓ_0- or ℓ_1-norm guided DL for the task of classification for two reasons: *(i)* it gives a perfect block structured sparse code at the expense of a stricter condition; and *(ii)* it could lead to potentially better performance when the dictionary size or the intrinsic dimension of data is small. Note that we have assumed a noiseless condition, which will be extended to the case of Gaussian noise in future work. We have also taken an inductive approach for analysis rather than analyzing the solution of the algorithm. In next section, we will demonstrate the performance of HiDL and GDDL using empirical results.



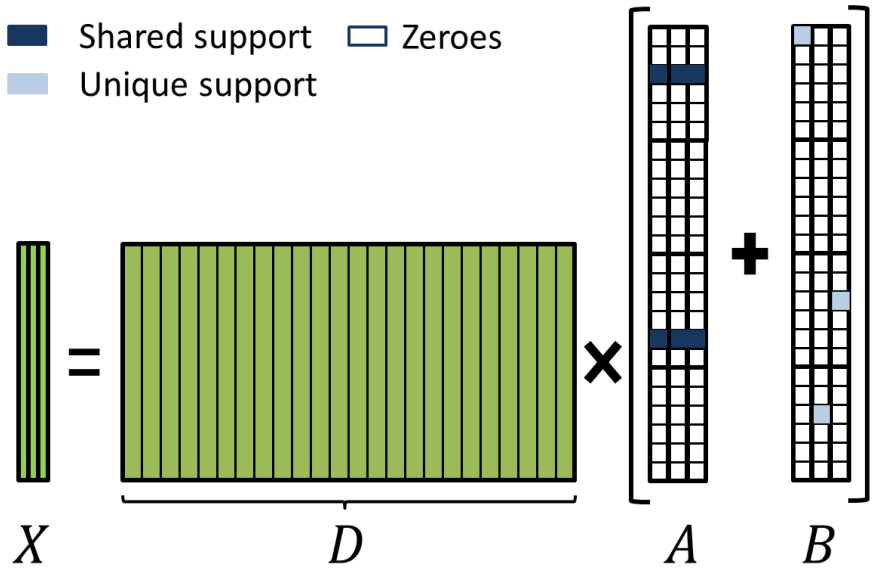
(a) ℓ_1-norm based DL

(b) locality based DL

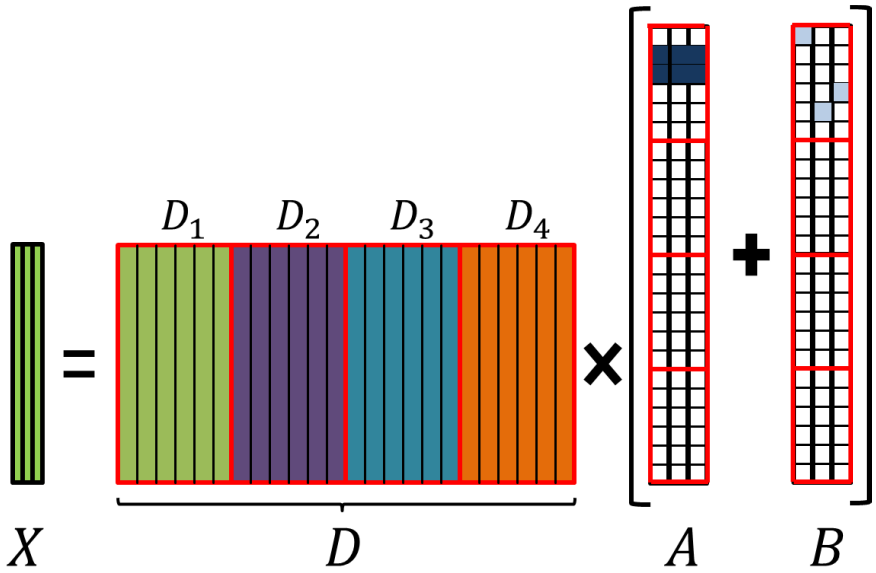
(c) proposed HiDL

(d) proposed GDDL

Figure 4.2: Comparison of proposed HiDL and GDDL approaches with other methods. Data matrix $\mathbf{X}$ are represented by grey circles and squares, corresponding to two different classes. The dictionary $\mathbf{D}$ lies on an oblique manifold.[1] Green and purple indicates selected dictionary atoms from different classes. Red dotted curve represents the boundary that separates sub-dictionaries of different classes. In (a), ℓ_1-norm based DL maps the data to a few dictionary atoms without limitation on their locations. In (b), the input is mapped to a few dictionary atoms in a certain neighborhood by locality constraint. However, data close to the class boundary could still be mapped to the dictionary atoms from wrong classes. In (c), HiDL forces the data to use a few atoms from same sub-dictionary (same class). In (d), GDDL separates the chosen atoms with the same label to two sub-groups: shared dictionary atoms (solid colored circle and square) and unique dictionary atoms (dashed colored circle and square).



(a) Dirty Model



(b) Group Structured Dirty Model (GSDM)

Figure 4.3: Comparison between the signal models of the Dirty Model and GSDM. Data $\mathbf{X}$ belongs to the same class. For the Dirty Model, the dictionary $\mathbf{D}$ only contains atoms for the same class while that of GSDM uses sub-dictionaries for four different classes, i.e., $\mathbf{D}_1, ..., \mathbf{D}_4$. The sparse coefficients $\mathbf{A}$ and $\mathbf{B}$ for GSDM are forced to capture the shared supports (dark blue) and unique supports (light blue) within the group boundary (red line), while the Dirty Model does not impose such constraint.

Chapter 5

Experimental Validation of Structured Dictionary Learning Methods

In this section, we compare the proposed HiDL and GDDL to various existing dictionary learning methods on synthetic dataset, real datasets for tasks of neural recording, face recognition, and object classification. The public datasets used in this section are the Leicester neural signal database,[47] Extended Yale B Face Database,[90] the AR Face Database,[91] and the Caltech101 Dataset.[92]

For neural recording case, the benchmark approaches are the proposed data dictionary in Chapter 3, trained dictionary,[43] wavelet dictionary,[2] Signal Dependent Neural Compressed Sensing Method (SDNCS),[43] DWT based CS method (DWT-CS),[2] and

Table 5.1: Objective functions of DL and classifiers used for different methods. Note that the last term in BGSC-ICS is an intra-block coherence suppression term and **Q** used in LC-KSVD is an ideal discriminative sparse code. For more details, readers could refer to the original papers.

Method	Objective function for DL	Classifier
K-SVD[16]	$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} \frac{1}{2}\|\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|\|_2^2 \; s.t. \; \|\|\mathbf{a}_i\|\|_0 \leq k$	linear
D-KSVD[27]	$\min_{\mathbf{D},\mathbf{A},\mathbf{W}} \sum_{i=1}^{N} \frac{1}{2}\|\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|\|_2^2 + \gamma\|\|\mathbf{L} - \mathbf{W}\mathbf{A}\|\|_F^2 + \beta\|\|\mathbf{W}\|\|_F^2 \; s.t. \; \|\|\mathbf{a}_i\|\|_0 \leq k$	linear
SRC[48]	Using training data directly	SRC
LLC[78]	$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} (\frac{1}{2}\|\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|\|_2^2 + \lambda_1\|\|\mathbf{e}_i \odot \mathbf{a}_i\|\|_2^2)$	linear SVM
BGSC-ICS[77]	$\min_{\mathbf{D},\mathbf{A}} \sum_{i=1}^{N} \frac{1}{2}\|\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|\|_2^2 + \lambda_3 \sum_{g\in\mathcal{G}} \|\|\mathbf{A}_{c,[g]}\|\|_F + \gamma \sum_{g\in\mathcal{G}} (\sum_{s,t\in g, s\neq t} \|\|\mathbf{d}_s^\top \mathbf{d}_t\|\|_2^2)$	(4.2.25)
LC-KSVD[28]	$\min_{\mathbf{D},\mathbf{A},\mathbf{W}} \sum_{i=1}^{N} \frac{1}{2}\|\|\mathbf{x}_i - \mathbf{D}\mathbf{a}_i\|\|_2^2 + \gamma\|\|\mathbf{L} - \mathbf{W}\mathbf{A}\|\|_F^2 + \beta\|\|\mathbf{Q} - \mathbf{A}\|\|_F^2 \; s.t. \; \|\|\mathbf{a}_i\|\|_0 \leq k$	linear
proposed HiDL	(4.2.5)	(4.2.25)
proposed GDDL	(4.2.12)	(4.2.25)

Transformation based method (on-chip DWT).[37]

For image classification cases, the benchmark algorithms are Sparse Representation-based Classification (SRC),[48] K-SVD,[16] Dictionary Learning with Structured Incoherence (DLSI),[30] Discriminative K-SVD (D-KSVD),[27] Locality-constrained Linear Coding (LLC),[78] Fisher Discrimination Dictionary Learning (FDDL),[29] Block and Group Regularized Sparse Modeling with Intra-block Coherence Suppression (BGSC-ICS),[77] Label Consistent K-SVD (LC-KSVD),[28] K-SVD[16] for dictionary learning of each class and HiLasso[10] for sparse coding (K-SVD + HiLasso), and K-SVD[16] for dictionary learning of each class and Dirty Model[31] for sparse coding (K-SVD + Dirty Model). A summary of DL objective functions and classifiers used in these benchmark methods is presented in Table 5.1.

We use SNDR, block coherence,[93] sparse code discrimination index (SDI), and classification accuracy for comparison purposes. The classification accuracy is defined as the percentage of correctly classified test data.

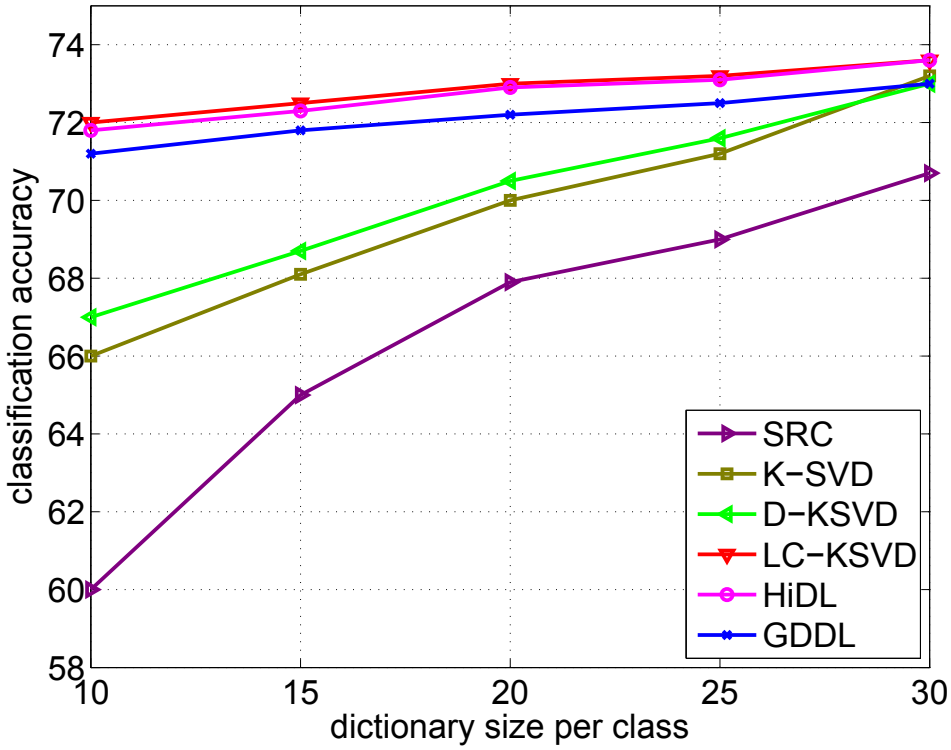


Figure 5.1: Effect of dictionary size on classification performance of different DL methods. For Caltech 101 dataset, the size of training samples per class is fixed to 30. The dictionary atoms per class is varied from 10 to 30. As can be seen, HiDL, GDDL and LC-KSVD outperforms SRC, K-SVD and D-KSVD. GDDL does not perform as well as HiDL because of the nature of the dataset. The benefit of adding hierarchical sparsity is especially helpful when the dictionary size is small.

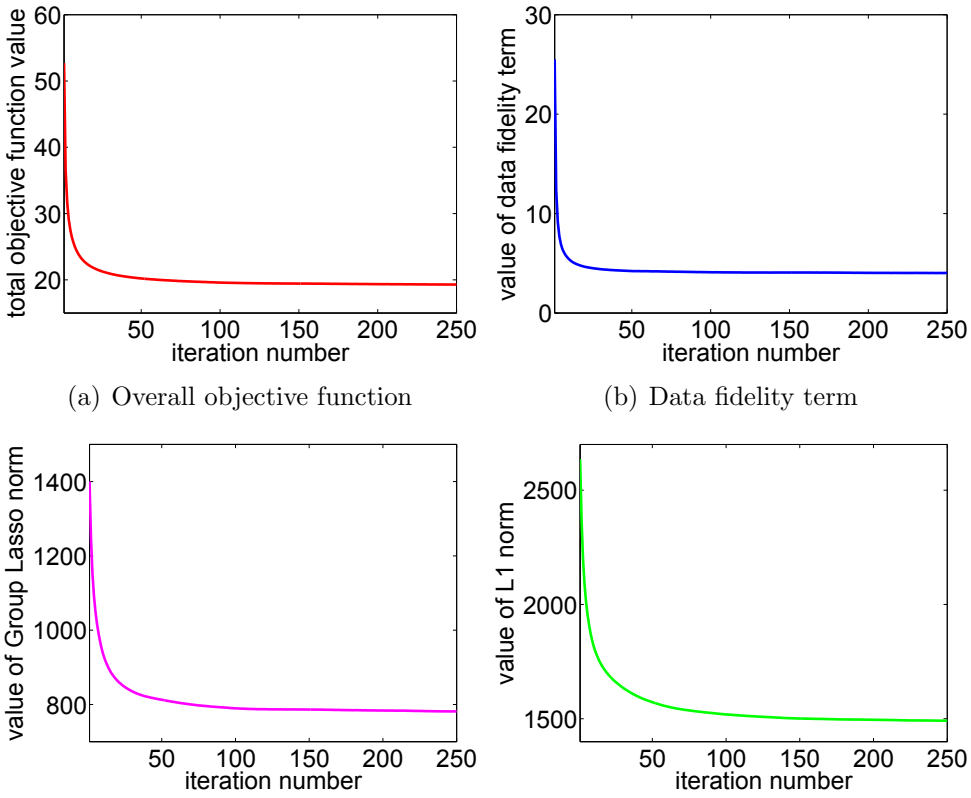
5.1 Parameter Selection

Dictionary Size: In all experiments, the initial dictionary for both HiDL and GDDL are random selections from training data. As shown in,[28,29] the larger the dictionary size is, the better classification performance it can generally yield. The drawback of a large dictionary size is that the size of problem becomes large simultaneously. Therefore, the ideal dictionary learning method is the one that can achieve a certain level of high performance using a small dictionary size. To compare the pro-

posed method with other approaches on this front, we use the Caltech101 Dataset as an example. For each class, we randomly choose 30 samples for training and the rest for testing. The number of dictionary atoms for each class varies from 10 to 30. As shown in Fig 5.1, all DL methods improve when the dictionary size becomes larger. Also, as proved in the previous section, our proposed HiDL and GDDL are comparable to LC-KSVD and all three methods consistently outperforms other sparsity driven approaches. This is consistent with our previous theoretical analysis. GDDL does not perform as well as HiDL for this dataset, probably because the dataset has very large within-class variability so the group structured dirty model does not fit the nature of the data. In contrast to other methods, HiDL and GDDL enforces label consistency implicitly using structured sparsity instead of adding extra constraint $f_{\mathbf{A}}(\cdot)$, therefore controlling the problem size.

Regularization Parameters: The choice of regularization parameters depends on the application and data. If a Bayesian approach is chosen for the sparse coding step, it will allow us to understand the connection between regularization parameters and data characteristics.[60] Here we adopt the convex optimization based approach, and we use 5-fold cross validation on training data to find the parameters that give the best results.

Stopping rule: The stopping rule for HiDL and GDDL could be such that either the change of objective function in (4.2.5) and (4.2.12) are small enough or the maximum iteration number has been reached. The objective function of both



(a) Overall objective function

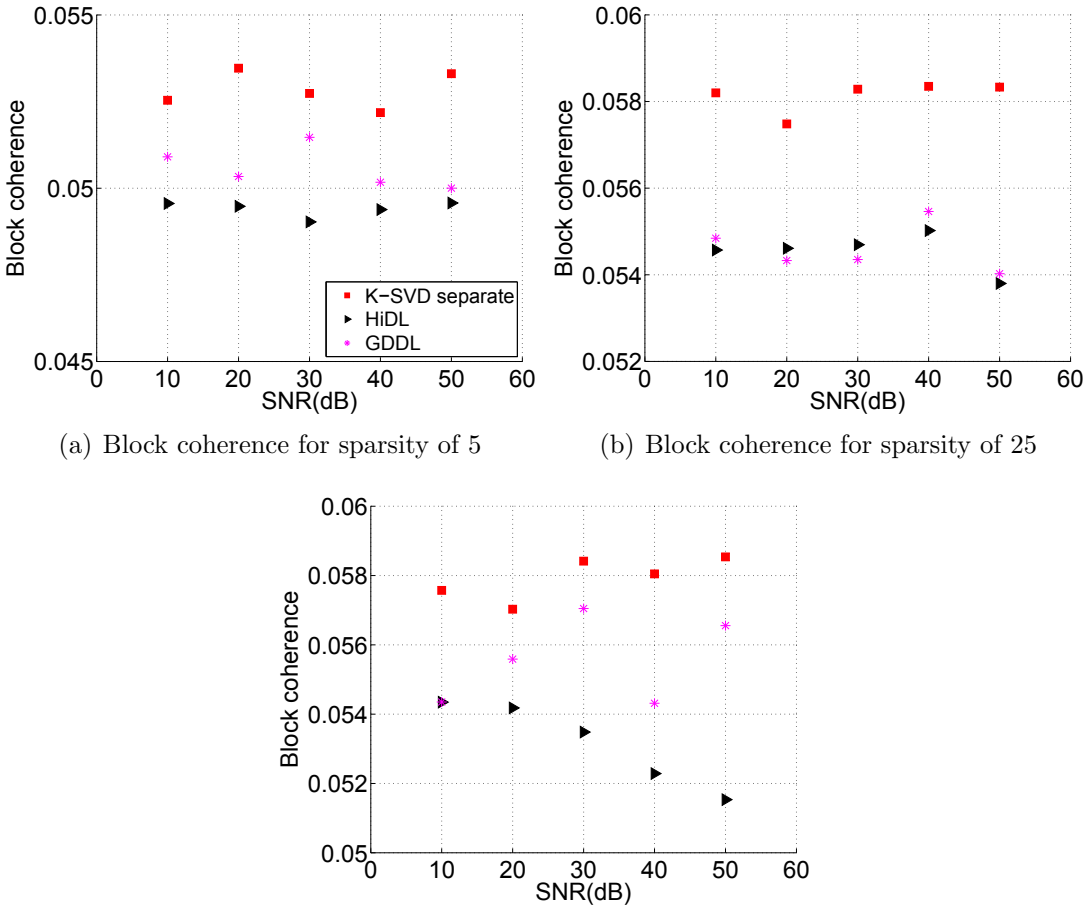
(b) Data fidelity term

(c) Regularization on $\mathbf{A}$

(d) Regularization on $\mathbf{B}$

Figure 5.2: Convergence of GDDL using the Extended Yale B dataset. The convergence of total objective function, the data fidelity term $||\mathbf{X} - \mathbf{DA} + \mathbf{B})||_F^2$, the regularization on $\mathbf{A}$ $\left(\sum_{c=1}^{C}(\lambda_1||\mathbf{A}_c||_{1,2} + \lambda_3 \sum_{g\in\mathcal{G}} ||\mathbf{A}_{c,[g]}||_F)\right)$ and the regularization on $\mathbf{B}$ $\left(\sum_{c=1}^{C}(\lambda_2||\mathbf{B}_c||_{1,1} + \lambda_4 \sum_{g\in\mathcal{G}} ||\mathbf{B}_{c,[g]}||_F)\right)$ are shown in (a), (b), (c) and (d), respectively.

HiDL and GDDL are non-convex, thus the proposed algorithm cannot find a global optimal solution. For the ℓ_1-norm regularized DL,[19] it is shown that a stationary point could be found if the sufficient condition for the uniqueness of sparse coding step is satisfied. In,[10,94] the authors also prove such condition for the HiLasso norm. Following similar methodology, we could potentially show that the proposed HiDL and GDDL do converge to a stationary point. The proof itself is beyond the scope of

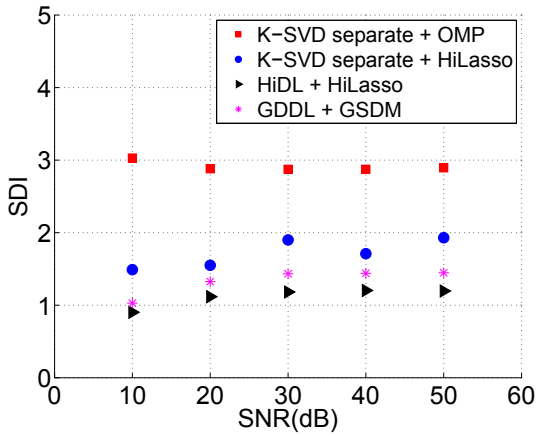


(a) Block coherence for sparsity of 5

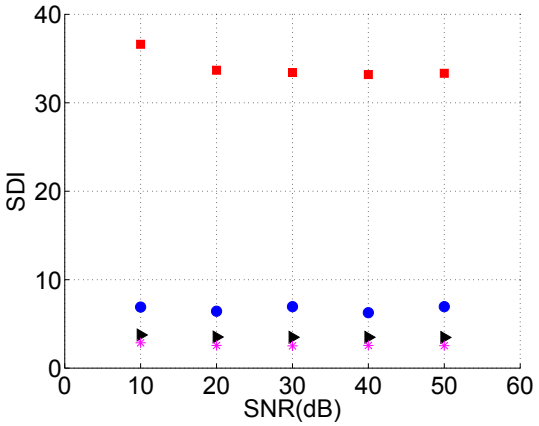
(b) Block coherence for sparsity of 25

(c) Block coherence for sparsity of 40

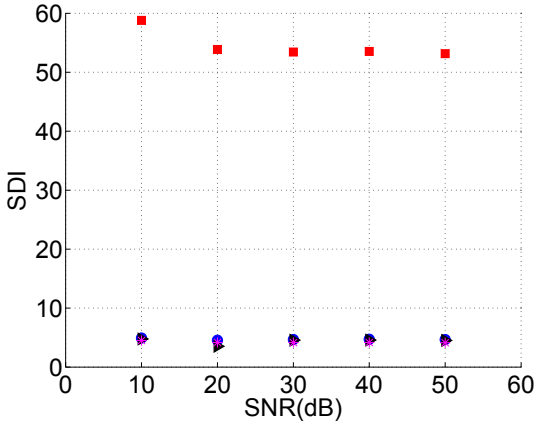
Figure 5.3: Comparison of block coherence using dictionaries learned from different approaches. Under different SNRs and sparsity ratios, the dictionaries generated by both HiDL and GDDL are more discriminative than *K-SVD separate*.



(a) SDI for sparsity of 5



(b) SDI for sparsity of 25



(c) SDI for sparsity of 40

Figure 5.4: Comparison of SDI using different dictionaries and sparse coding approaches. Under different SNRs and sparsity ratios, the sparse codes generated by both *HiDL + HiLasso* and *GDDL + GSDM* are more discriminative than that of *K-SVD separate + OMP* and *K-SVD separate + HiLasso*.

this paper and will be presented in our future work. Here, we only show empirically the change of the objective function using Extended Yale B dataset. As shown in Fig 5.2 for GDDL, the value of the whole objective function in (4.2.12), the data fidelity term, the $\ell_{1,2}$-norm and collaborative Group Lasso norm, and the $\ell_{1,1}$-norm and collaborative Group Lasso norm converge around 100 iterations. The experiment setup will be described in Section 5.5.

5.2 Synthetic Dataset

Unlike reconstruction-oriented dictionary learning, the GDDL framework is geared towards the task for classification. The proposed HiDL and GDDL use the group structure $\mathcal{G}$ to enforce the label consistency between sub-dictionaries and training data. Such mapping could also be realized by training a sub-dictionary $\mathbf{D}_c$ ($c = 1, ..., C$) for each class independently and then concatenating the sub-dictionaries to build $\mathbf{D} = [\mathbf{D}_1, ..., \mathbf{D}_C]$. To understand the benefit of incorporating the structured sparsity in the DL process, we use block coherence[93] to compare the dictionaries learned by HiDL and GDDL with concatenating K-SVD dictionaries of different classes (as in Fig 5.3). For simplicity, we refer to the latter approach as *K-SVD separate*. We also investigate into the discriminative power of the sparse coefficients (as in Fig 5.4). Specifically, we compare sparse coefficients of the same dataset generated by four different approaches: proposed HiDL dictionary with HiLasso sparse coding,[10]

proposed GDDL with GSDM, K-SVD dictionary with Orthogonal Matching Pursuit (OMP),[95] and K-SVD dictionary with HiLasso sparse coding.[10] For simplicity, we refer to the last two approaches as *K-SVD separate + OMP* and *K-SVD separate + HiLasso*, respectively.

Experiment Setup: We generate the synthetic data under different sparsity setting and signal-to-noise ratio (SNR) levels. The true sub-dictionaries $\mathbf{D}_c$ are generated for 10 different classes ($C = 10$). Each sub-dictionary is a 20 by 50 random Gaussian matrix with unit ℓ_2-norm for each column. Therefore, the group structure $\mathcal{G}$ is 10 groups with 50 sub-dictionary atoms in each group. For each class, the data $\mathbf{x}_i (i = 1, ..., 1500)$ is a random combination of dictionary atoms from the same sub-dictionary while the values of $\mathbf{a}_i$ are drawn from a random Gaussian distribution with zero mean and unit standard deviation. The sparsity of $\mathbf{a}_i$ are set to 5, 25 and 40 to simulate different levels of within-group sparsity. When the sparsity is 5, the within-group variation is more prominent while the within-group similarity is more significant when sparsity is 40. By concatenating data from all 10 classes, the data matrix $\mathbf{X}$ is of dimension 20 by 15000. Furthermore, zero-mean Gaussian noise is added to the data so that the SNR ranges from 10 to 50dB. Under each noise level, the experiment is repeated 10 times and each time the data is randomly splitting into two halves, training and test set.

The input parameters of sparsity for *K-SVD separate* are set to the true values. For HiDL and GDDL, all regularization parameters are set to 0.1, 0.05 and 0.01 for

each of three sparsity levels, respectively.

Criteria: A dictionary is more discriminative if the right atoms are more likely to be found by sparse coding. According to,[93] a smaller block-coherence will lead to a higher probability for sparse coding to find the right block (sub-dictionary), thus make the dictionary more discriminative. Therefore, to quantify how discriminative the trained dictionary is, we adopt the concept of block-coherence from,[93] which is,

$$\mu_B = \max_{c_1,c_2\in\{1,\ldots,C\},c_1\neq c_2} \frac{1}{K_c}\rho(\mathbf{D}_{c_1}^\top \mathbf{D}_{c_2}) \tag{5.2.1}$$

where sub-dictionaries $\mathbf{D}_{c_1}$ and $\mathbf{D}_{c_2}$ are for class c_1 and c_2 with equal block size K_c. The spectral norm $\rho(\cdot)$ of $\mathbf{D}_{c_1}^\top \mathbf{D}_{c_2}$ is the square root of the largest eigenvalue of the matrix product.

A sparse code is more discriminative if the sparse codes for signals of same class are more similar while that for different classes are more different. Therefore, we use Fisher discrimination criterion[29] to measure the discriminatory power of the sparse code for both training and test data, which is defined as sparse code discrimination index (SDI):

$$SDI = \frac{1}{N}\left[tr(\mathbf{S}_{within}(\mathbf{A})) - tr(\mathbf{S}_{between}(\mathbf{A}))\right]. \tag{5.2.2}$$

The with-in cluster scatter measure $\mathbf{S}_{within}(\mathbf{A})$ is defined as

$$\mathbf{S}_{within}(\mathbf{A}) = \sum_{c=1}^{C} \sum_{\mathbf{a}_i\in\mathbf{A}_c} (\mathbf{a}_i - \mathbf{m}_c)(\mathbf{a}_i - \mathbf{m}_c)^\top$$

where $\mathbf{A}_c$ is the sub-matrix formed by extracting the columns in $\mathbf{A}$ that corresponds to the c-th class. Here, $\mathbf{m}_c$ is the mean column vector of $\mathbf{A}_c$. The between-class scatter $\mathbf{S}_{between}(\mathbf{A})$ can be calculated by:

$$\mathbf{S}_{between}(\mathbf{A}) = \sum_{c=1}^{C} N_c(\mathbf{m}_c - \mathbf{m})(\mathbf{m}_c - \mathbf{m})^\top$$

where $\mathbf{m}$ is the mean column vector of $\mathbf{A}$ and N_c is the number of signal in c-th class. A smaller SDI indicates a smaller within-class scatter and a larger between-class scatter, thus corresponding to a more discriminative sparse code. Notice that for GDDL, we only use the sparse coefficient $\mathbf{A}$ corresponding to the shared support to calculate SDI, which is also what we use for classification.

Remark: The simulation results are shown in Fig 5.3 and Fig 5.4. As can be seen from Fig 5.3, HiDL and GDDL consistently generate dictionaries with smaller block-coherence than *K-SVD separate*. GDDL dictionary has a higher block coherence than HiDL, except when sparsity is 25. This is probably because when sparsity is 25 (50% of the sub-dictionary size), the within-group variation and within-group similarity is balanced and the GDDL model suits the data better than HiDL.

The sparse code of the training data is also found with respect to the learned dictionary and the corresponding SDI is calculated using (5.2.2). For different in-group sparsity and SNR levels, the SDI for both HiDL and GDDL are consistently smaller compared to that of either *K-SVD separate + OMP* or *K-SVD separate +*

HiLasso. Notice that the when the sparsity grows, the SDI grows as well. However, the change of SDI for HiDL, GDDL, and *K-SVD separate + HiLasso* are not as much as that of *K-SVD separate + OMP*, which demonstrates the consistent discriminative power of structured sparsity. We see that the SDI for *K-SVD separate + HiLasso* gets closer to the level of HiDL and GDDL as the sparsity becomes larger. This could suggests that when the sparsity increases, having a structured dictionary learning approach is not as important as having a structured sparse coding approach. We also notice that GDDL works especially well when the within-group variation and within-group similarity is balanced in some extent (Fig 5.4(b)), but not as well when the within-group variation or the within-group similarity is high (Fig 5.4(a) and (c)), respectively. The results are similar for the testing data and therefore omitted. We will try to understand these phenomena from a theoretical standpoint in our future work.

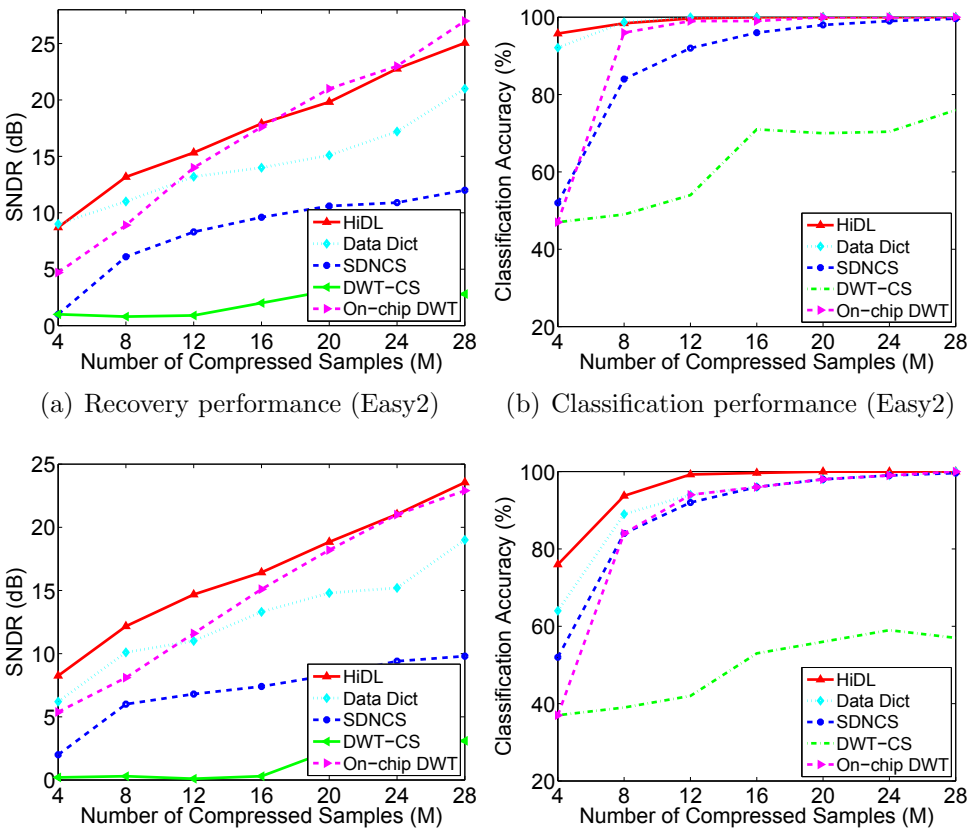
In summary, HiDL and GDDL have the advantage of forcing the sub-dictionaries for different classes to compete against each other in the sparse coding step and only the 'winners' get updated in the following dictionary update stage. Furthermore, the group structure $\mathcal{G}$ could ideally restrict the sparse codes for different classes to live in different subspaces, therefore also improving the discriminative power of the sparse codes. As pointed out in Section II.A, structured sparsity incorporating the sparsity, locality and grouping can lead to a more discriminative dictionary as HiDL and GDDL do.

5.3 Neural Recording

We use the Leicester neural signal database,[47] which contains 20 simulation datasets. Each dataset contains spikes from three different types of neurons with different noise levels. The datasets are named by the difficulty to perform spike sorting, such as Leicester Difficult1, Difficult2, Easy1, and Easy2.

5.3.1 Performance of HiDL

We randomly split the data into 20% for training and 80% for testing. The reconstruction and classification results for Leicester - Difficult1 and Easy2 with 0.005 noise std are shown in Fig. 5.5. We use the same wavelet based classifier (WLC) as in.[43] It can be seen that the proposed HiDL is more effective (2 dB) in recovery than our previously proposed data dictionary as well as other benchmark methods. On par with the on-chip DWT method, it has the benefit of a simple on-chip implementation while we incorporate more prior information for off-chip processing. HiDL also demonstrates superior classification performance, especially at the low CR range (e.g., measurements $M = 4$). It achieves a 3% improvement of classification accuracy when the neural signals are quite different among each class (e.g., Easy2 dataset) and more than 10% improvement when the signals are more similar to each other (e.g., Difficult2 dataset). This again proves the benefit of incorporating the label information as prior knowledge into the dictionary learning process.



(a) Recovery performance (Easy2) (b) Classification performance (Easy2)

(c) Recovery performance (Difficult1) (d) Classification performance (Difficult1)

Figure 5.5: Performance comparison between proposed HiDL and other CS approaches.

5.3.2 Performance of HiDL-CS

Furthermore, we evaluate the performance of proposed HiDL-CS both qualitatively and quantitatively. Using Easy2 and Difficult2 datasets from the Leicester neural signal database, we follow the same setup as in previous section and compare the performance of dictionaries learned by HiDL, HiDL-CS and fixed DWT transformation. As seen in Fig. 5.6, we show at different Compression Ratio (CR), the recovery performance of learned dictionaries. It can be seen that HiDL has superior recovery performance (e.g., above 10 dB) and improves dramatically when we acquire more measurements. For HiDL-CS, the recovery result looks very noisy when the CR is as low as 10% due to the loss of information during the compression process. As CR becomes higher, the recovered signal starts to present the shape of the desired signal and gradually capture the geometric features of the signal while still having some artifacts. Nevertheless, HiDL-CS consistently outperforms DWT dictionary for its recovery performance. Furthermore, we compare the performance of different dictionaries at different CRs as shown in Fig. 5.7. As for classification, HiDL-CS outperforms, particularly in the low CR ranges (5% - 35%). Thus, HiDL-CS could be used as a viable alternative for training a dictionary with good enough recovery performance and excellent classification performance without the need of training the dictionary at full acquisition rate. This enables the potential of having a continuous online dictionary learning system for neural recording.

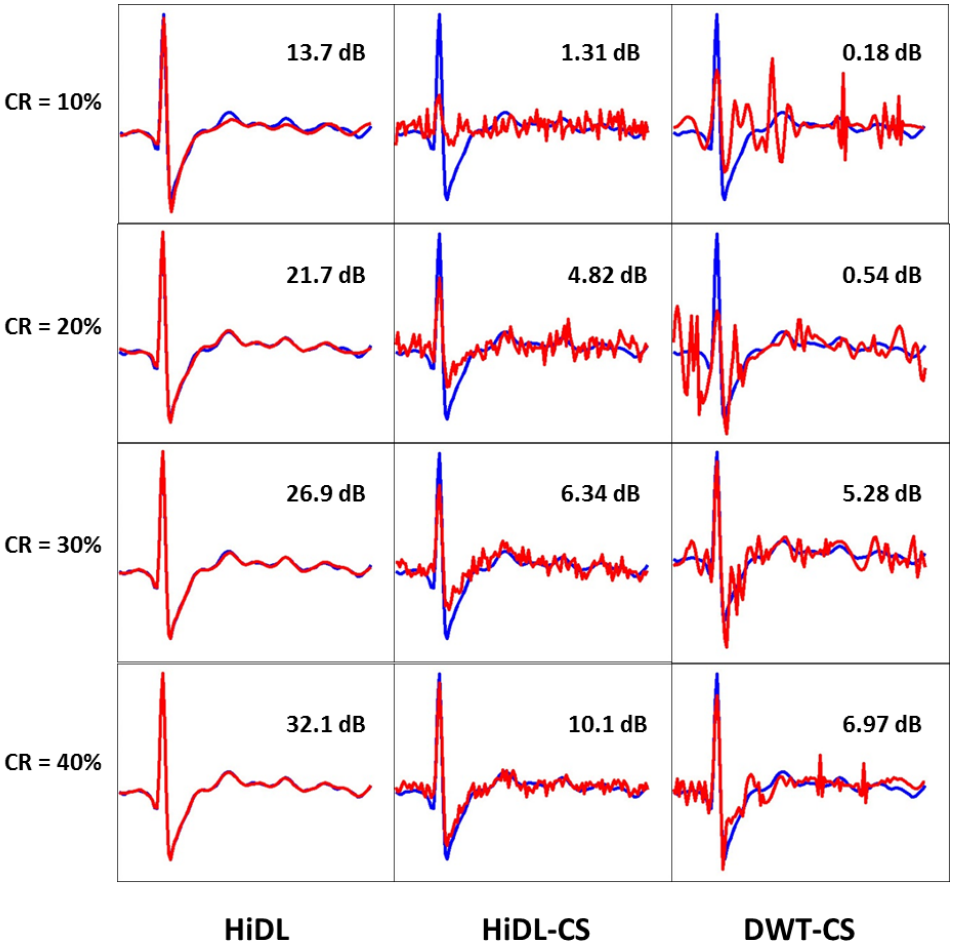
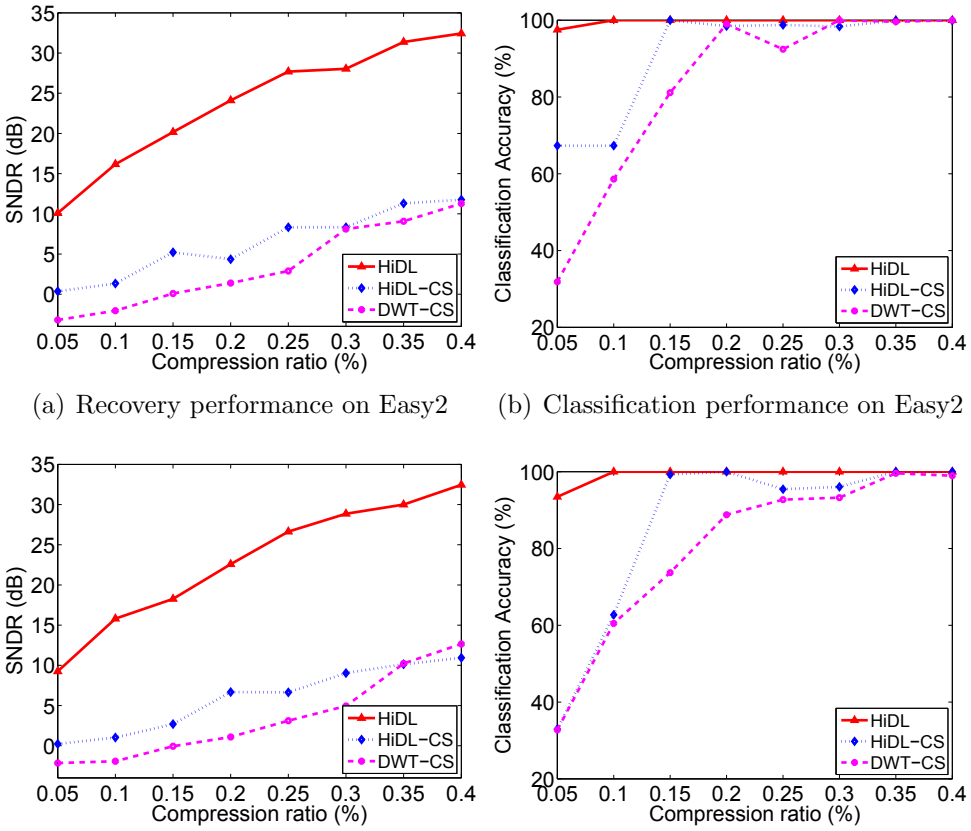


Figure 5.6: Recovery results of a single spike using different dictionary choices at different CRs. The recovery results are measured using SNDR (dB). The groundtruth is plotted in blue and the recovered signal is plotted in red.



(a) Recovery performance on Easy2

(b) Classification performance on Easy2

(c) Recovery performance on Difficult2

(d) Classification performance on Difficult2

Figure 5.7: Performance comparison between proposed HiDL-CS and other dictionary learning approaches.

5.3.3 Performance of GDDL for Spike Co-occurrence case

For the real case, we might have two or more neurons firing at the same time, thus a signal frame may contain two or more spikes, which are super-positioned on top of each other, resulting in a large spike. In our previous work,[43] an on-chip detection curcuit is used to relay these non-ideal information to the recovery algorithm. Using multiple thresholds, the on-chip system could signal the off-chip recovery algorithms to attempt to recover these signal frames by using two different dictionaries, belonging to two different spike classes. However, this is by assuming that the training data itself does not contain such contamination. To address this issue of contamination of the training data, we apply the proposed GDDL to learn a dictionary that can separate the with-in class similarity (target neuron signals) from the with-in class difference (randomly firing non-target neurons). To build a simulated dataset, we use Easy1 dataset as the target neuron signals and Difficult1, Difficult2 and Easy 2 datasets as the non-target neuron signals. To generate the randomly firing non-target neurons, we randomly choose it from these datasets, multiply with a random weight following a zero mean Gaussian distribution and add to the target neuron signals. Since the recovery performance in terms of SNDR could be misleading, we use only classification accuracy to measure the performance as shown in Table 5.2. It can be seen that GDDL can have higher tolerance to such un-desired noise (non-

Table 5.2: Comparison of proposed GDDL and other state-of-art DL methods for spike mixing case. The best results are achieved by GDDL and bolded.

CR	5%	15 %	25 %
GDDL	**84.5**	**91.9**	**92.3**
HiDL	81.2	89.5	90.9
Data Dictionary	79.0	84.4	86.3
SDNCS	74.7	78.1	80.2
DWT-CS	40.3	48.2	52.4

target neuron signals) and drive for a higher classification performance. Although HiDL performs well in the ideal case, the mixed signal can complicate the learned dictionary, thus deteriorate the classification results.

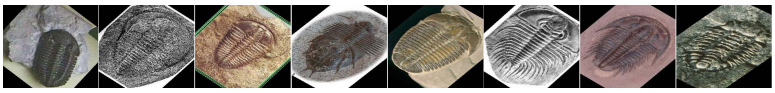
5.4 Object Classification

The Caltech 101 dataset contains 9,144 images in 102 categories, including animals, cars, planes, etc. Each category has 40 to 800 images, with most categories having around 50 images. Pictures from same class have drastic shape variability and the spatial pyramid features[96] are used as the input signal, which is same as.[28,78] The dimension of each feature is 3000. The size of the dictionary is the same as the number of training samples per class. We vary the number of training samples per class from 10 to 30. The experiments are repeated 10 times while HiDL and GDDL are compared with K-SVD, D-KSVD, SRC, LLC, BGSC-ICS, LC-KSVD, K-SVD + HiLasso, and K-SVD + Dirty Model. Results are shown in Table 5.3 with some results

(a) accordion

(b) car

(c) motorbikes



(d) trilobite

Figure 5.8: Examples of categories in Caltech 101 that achieve 100% classification accuracy by HiDL.

being reported by.[28] The best results of BGSC-ICS are reproduced using the code provided by the authors. For K-SVD + Dirty Model approach, we use K-SVD and Dirty Model on each class separately and then use a linear classifier. Our proposed HiDL consistently outperforms other approaches. As pointed out early, our proposed GDDL does not perform as well as HiDL probably because this particular dataset has large within-class variability. However, it is shown later that for face datasets, GDDL outperforms HiDL. Several of the object classes that achieve 100% accuracy by HiDL are shown in Fig 5.8. The regularization parameters for HiDL are 0.009 and 0.007 and those for GDDL are 0.005, 0.004, 0.004 and 0.007, respectively.

Table 5.3: Comparison of proposed HiDL and GDDL and other state-of-art DL methods using Caltech 101 dataset. The dictionary size of each class is the same as the training samples per class. The best results are achieved by HiDL and bolded.

Training data size per class	10	15	20	25	30
K-SVD	59.8	65.2	68.7	71.0	73.2
D-KSVD	59.5	65.1	68.6	71.1	73.0
SRC	60.1	64.9	67.7	69.2	70.7
LLC	59.77	65.43	67.74	70.16	73.44
BGSC-ICS	62.4	67.5	69.4	71.6	73.3
LC-KSVD	63.1	67.7	70.5	72.3	**73.6**
K-SVD + HiLasso	61.9	66.5	69.2	70.7	72.3
K-SVD + Dirty Model	60.2	64.6	68.3	69.9	70.8
HiDL	**63.4**	**68.1**	**70.9**	**72.7**	**73.6**
GDDL	62.1	66.3	69.0	71.0	73.1

5.5 Face Recognition

Face recognition is an important category of image classification tasks with applications in video surveillance and mobile imaging. The two most widely used face recognition dataset are Extended Yale B database and AR databse. Captured under various lighting conditions, the Extended Yale B database consists of 2,414 frontal-face images for 38 individuals (around 64 images per person). Similarly, the AR database has over 4,000 frontal-face images for 126 individuals, which are also taken under different conditions, including facial expressions, lighting conditions, and occlusions. Same as,[28,48] we crop the Extended Yale B images to the dimension of 192 $\times$ 168 pixels, normalized and projected to a vector of dimension 504 using random Gaussian projection. The AR dataset is cropped to the dimension of 165 $\times$ 120 pix-

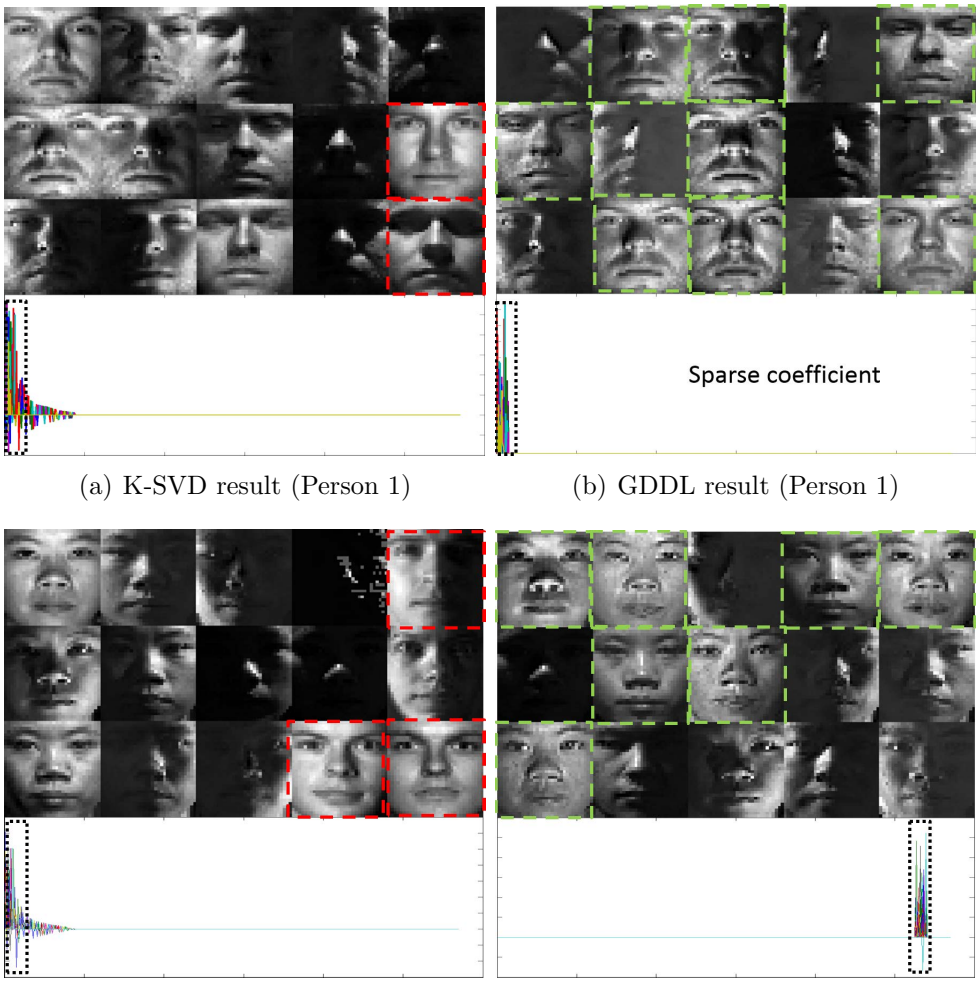
els, normalized and projected to a vector of dimension 540 using random Gaussian projection. For Extended Yale B, we randomly select half of the images for training and the other half for testing in each class. For each class in the AR dataset, twenty images and six images are randomly selected for training and testing, respectively. The dictionary size for Extended Yale B and AR dataset is 15 and 5 dictionary atoms for each class, respectively. Therefore, the total dictionary contains 570 and 500 atoms. The experiment is carried out 10 times with different randomly chosen partitions. The regularization parameters for HiDL are 0.01 and 0.005 and the regularization parameters for GDDL are 0.01, 0.009, 0.005 and 0.006, respectively. The average classification accuracy is again compared with D-KSVD, LLC, BGSC-ICS, LC-KSVD, K-SVD + HiLasso, and K-SVD + Dirty Model in Table 5.4. The performances of benchmark algorithms are as reported by,[28] which have been tuned to achieve the best results. The best results of BGSC-ICS are reproduced using the code provided by the authors.

The proposed HiDL and GDDL achieve an improvement of more than 3 percentage units in terms of classification accuracy using the same dictionary size for both datasets. To further demonstrate the difference between structured sparsity (i.e., GDDL) and the ℓ_0-norm (K-SVD) in DL, the learned dictionary and the sparse code for Person 1 and 36 of Yale B dataset are presented in Fig 5.9. Note that the sparse code shown here is that of all training data in each class. The K-SVD dictionary for each class is chosen by finding the dictionary atoms that have the largest magnitude

Table 5.4: Comparison of proposed HiDL and GDDL with other state-of-art DL methods on face recognition tasks. All methods use the same dictionary size. The best results are achieved by proposed HiDL and GDDL.

Method	D-KSVD	LLC	BGSC-ICS	LC-KSVD	HiDL	GDDL
Extended Yale B	94.1	90.7	96.5	95.0	**98.0**	**98.2**
AR	88.8	88.7	93.2	93.7	**96.4**	**96.7**

of sparse coefficients. We can see that the K-SVD dictionary has mixed some similar faces from other classes into the desired class (red dotted). Simultaneously, the corresponding sparse code for training data in the same class has a longer-tail distribution outside the group index (Fig 5.9 (a) and (c)). In contrast, the dictionary learned by GDDL guarantees the dictionary atoms in the group index having the same label. And the sparse code of all training data in this class is strictly within the group index, which justifies our motivation as explained in Fig 4.2. Moreover, the dictionary atoms corresponding to the GDDL's shared supports (green dotted figures in GDDL dictionaries) capture the similarity between data in the same class while those corresponding to unique supports (un-dotted figures in GDDL dictionaries) indicate the within-class variation.



(a) K-SVD result (Person 1) (b) GDDL result (Person 1)

(c) K-SVD result (Person 36) (d) GDDL result (Person 36)

Figure 5.9: The learned dictionary and the sparse coefficient of training data using K-SVD and GDDL. The sparse codes for all training data in the same class are ploted in the bottom. It can be observed that the labels of dictionary atoms learned by GDDL are consistent while K-SVD can mix the similar faces (red dotted figures). The sparse code for training data indicates that the proposed method can strictly enforce the correct group be chosen while K-SVD fails to do so. Moreover, the dictionary atoms corresponding to the GDDL's shared supports (green dotted figures) capture the similarity between data in the same class while those corresponding to unique supports (un-dotted figures) indicate the within-class variation.

Chapter 6

Summary and Future Work

First, we have presented in this thesis an energy-efficient multi-mode CS system for implantable neural recordings. Using data directly as the dictionary and a two-stage sensing strategy, our design is suited with simple circuit design and power efficiency. Moreover, we proposed two new working modes to leverage on the power of sparse representation to restore the full signal from only spike detection results or CS measurements of spike detection results. This provides an all-in-one device with a higher CR and energy efficiency. The same framework has also been tested for the case of Tetrode CS by extending our recovery algorithm to the case of joint sparsity. Experiments on simulation and real datasets have demonstrated that the proposed framework outperforms other approaches and can guarantee energy efficiency, implementation simplicity and system flexibility all at once. Although we only demonstrate our framework using neural signal, it certainly can be applied to other biological sig-

nals (i.e., ECG) as well. In the future, we will investigate in three different directions for MEA CS: *(i)* a mechanism to dynamically allocate sensing channels among different electrodes; *(ii)* customizing our dictionary design and recovery algorithm for discriminative applications, such as spike sorting; and *(iii)* testing the performance of our system by conducting in-vivo experiments.

Second, we incorporate structured sparsity in the DL process for classification purposes. The proposed GDDL framework (including its single task version − HiDL and compressed sensing version − HiDL-CS) has two advantages compared to ℓ_0-and ℓ_1-norm regularized methods: *(i)* the dictionary atoms with same group index have same consistent label and this label consistency also exists between dictionary and training data; and *(ii)* the classification performance is more robust to small dictionary size or limited training data, providing computation benefits. Through synthetic and real datasets, we demonstrate that the HiDL and GDDL can generate more discriminative sparse codes, thus improve classification performance. We provide the conditions for HiDL to achieve optimal performance and show the theoretical advantage of HiDL to ℓ_1-norm regularized DL for classification tasks. In the future, we will focus on the theoretical analysis of the convergence and locality properties of the proposed HiDL and GDDL. Another interesting direction is to explore the case when the structure is unknown and to incorporate the learning of structure within the DL process automatically and systematically.

Finally, we demonstrate the performance of our proposed HiDL, HiDL-CS and

GDDL methods using the case of neural recording. It has been shown that HiDL could help guarantee recovery performance as well as classification accuracy at very low compression rario (5%). By extending HiDL to learn the dictionary from compressed measurements, we develop HiDL-CS which can yield comparable classification performance with satisfying recovery performance at low compression ratio (5%). To deal with the non-ideal case of having sparsely firing neuron occurring with target neurons in the training and testing data, we use GDDL to separate the target neurons (with-in class similarity) form the sparsely firing neurons (with-in class difference), which can generate great classification results and overcome the interference of the undesired neuron signals.

Besides previously mentioned extensions of related work, we are also interested in the case where data sensing is performed simultaneously from multiple co-located sources/sensors, yet within the same spatial-temporal neighborhood, recording the same physical event. In which case, newer highly non-stationary data is coming in all the time while older data constantly becomes outdated and less valuable due to either storage shortage or becoming irrelevant due to changes in signal statistics. Optimal prediction, sampling, representation, and estimation of these constantly-changing signals might require locally-adaptive representation that can quickly capture the signal characteristics within a small local neighborhood.

Bibliography

[1] P.-A. Absil, R. Mahony, and R. Sepulchre, *"Optimization algorithms on matrix manifolds"*. Princeton University Press, 2009.

[2] F. Chen, A. P. Chandrakasan, and V. M. Stojanovic, "Design and analysis of a hardware-efficient compressed sensing architecture for data compression in wireless sensors," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 3, pp. 744–756, 2012.

[3] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[4] S. Mallat, *"A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way"*. Academic Press, 2008.

[5] E. J. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 489–509, 2006.

[6] D. L. Donoho, "Compressed sensing," *IEEE Transactions on Information Theory*, vol. 52, no. 4, pp. 1289–1306, 2006.

[7] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[8] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2005.

[9] L. Jacob, G. Obozinski, and J. Vert, "Group lasso with overlap and graph lasso," *International Conference on Machine Learning (ICML)*, pp. 433–440, 2009.

[10] P. Sprechmann, I. Ramirez, G. Sapiro, and Y. C. Eldar, "C-hilasso: A collaborative hierarchical sparse modeling framework," *IEEE Transactions on Signal Processing*, vol. 59, no. 9, pp. 4183–4198, 2011.

[11] E. v. d. Berg and M. P. Friedlander, "Joint-sparse recovery from multiple measurements," *arXiv preprint arXiv:0904.2051*, 2009.

[12] E. G. Allstot, A. Y. Chen, A. M. Dixon, D. Gangopadhyay, H. Mitsuda, and D. J. Allstot, "Compressed sensing of ecg bio-signals using one-bit measurement matrices," *New Circuits and Systems Conference (NEWCAS)*, pp. 213–216, 2011.

[13] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.

[14] M. Elad, "Optimized projections for compressed sensing," *IEEE Transactions on Signal Processing*, vol. 55, no. 12, pp. 5695–5702, 2007.

[15] J. M. Duarte-Carvajalino and G. Sapiro, "Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization," *IEEE Transactions on Image Processing*, vol. 18, no. 7, pp. 1395–1408, 2009.

[16] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[17] K. Engan, K. Skretting, and J. H. Husøy, "Family of iterative ls-based dictionary learning algorithms, ils-dla, for sparse signal representation," *IEEE Transactions on Signal Processing*, vol. 17, no. 1, pp. 32–49, 2007.

[18] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," *Advances in Neural Information Processing Systems (NIPS)*, pp. 801–808, 2006.

[19] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online dictionary learning for sparse coding," *International Conference on Machine Learning (ICML)*, pp. 689–696, 2009.

[20] I. Ramírez, Ignacio, and G. Sapiro., "An mdl framework for sparse coding and

dictionary learning," *IEEE Transactions on Signal Processing*, vol. 60, no. 6, pp. 2913–2927, 2012.

[21] K. Yu, T. Zhang, and Y. Gong, "Nonlinear learning using local coordinate coding," *Advances in Neural Information Processing Systems (NIPS)*, pp. 2223–2231, 2009.

[22] R. Jenatton, J. Mairal, F. R. Bach, and G. R. Obozinski, "Proximal methods for sparse hierarchical dictionary learning," *International Conference on Machine Learning (ICML)*, pp. 487–494, 2010.

[23] K. Rosenblum, L. Zelnik-Manor, and Y. C. Eldar, "Dictionary optimization for block-sparse representations," *AAAI Symposium on Manifold Learning*, pp. 50–58, 2010.

[24] M. Zhou, H. Chen, J. Paisley, L. Ren, L. Li, Z. Xing, D. Dunson, G. Sapiro, and L. Carin, "Nonparametric bayesian dictionary learning for analysis of noisy and incomplete images," *IEEE Transactions on Image Processing*, vol. 21, no. 1, pp. 130–144, 2012.

[25] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2008.

[26] J. Mairal, F. R. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 4, pp. 791–804, 2012.

[27] Q. Zhang and B. Li, "Discriminative k-svd for dictionary learning in face recognition," *Computer Vision and Pattern Recognition (CVPR)*, pp. 2691–2698, 2010.

[28] Z. Jiang, Z. Lin, and L. S. Davis, "Learning a discriminative dictionary for sparse coding via label consistent k-svd," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1697–1704, 2011.

[29] M. Yang, a. X. F. L. Zhang, and D. Zhang, "Fisher discrimination dictionary learning for sparse representation," *International Conference on Computer Vision (ICCV)*, pp. 543–550, 2011.

[30] I. Ramírez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," *Computer Vision and Pattern Recognition (CVPR)*, pp. 3501–3508, 2010.

[31] A. Jalali, S. Sanghavi, C. Ruan, and P. K. Ravikumar, "A dirty model for multi-task learning," *Advances in Neural Information Processing Systems (NIPS)*, pp. 964–972, 2010.

[32] A. Rodriguez-Perez, J. Ruiz-Amaya, M. Delgado-Restituto, and A. Rodriguez-Vazquez, "A low-power programmable neural spike detection channel with em-

bedded calibration and data compression," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 2, pp. 87–100, 2012.

[33] S. Mitra, J. Putzeys, F. Battaglia, C. M. Lopez, M. Welkenhuysen, C. Pennartz, C. Van Hoof, and R. F. Yazicioglu, "24-channel dual-band wireless neural recorder with activity-dependent power consumption," *International Solid-State Circuits Conference (ISSCC)*, pp. 292–293, 2013.

[34] M. S. Chae, Z. Yang, M. R. Yuce, L. Hoang, and W. Liu, "A 128-channel 6 mw wireless neural recording ic with spike feature extraction and uwb transmitter," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 17, no. 4, pp. 312–321, 2009.

[35] B. Gosselin and M. Sawan, "An ultra low-power cmos automatic action potential detector," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 17, no. 4, pp. 346–353, 2009.

[36] B. Gosselin, A. E. Ayoub, J.-F. Roy, M. Sawan, F. Lepore, A. Chaudhuri, and D. Guitton, "A mixed-signal multichip neural recording interface with bandwidth reduction," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 3, no. 3, pp. 129–141, 2009.

[37] K. G. Oweiss, A. Mason, Y. Suhail, A. M. Kamboh, and K. E. Thomson, "A scalable wavelet transform vlsi architecture for real-time signal processing in

high-density intra-cortical implants," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 6, pp. 1266–1278, 2007.

[38] A. M. Kamboh, A. Mason, and K. G. Oweiss, "Analysis of lifting and b-spline dwt implementations for implantable neuroprosthetics," *Journal of Signal Processing Systems*, vol. 52, no. 3, pp. 249–261, 2008.

[39] H. Mamaghanian, N. Khaled, D. Atienza, and P. Vandergheynst, "Compressed sensing for real-time energy-efficient ecg compression on wireless body sensor nodes," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 9, pp. 2456–2466, 2011.

[40] A. M. Dixon, E. G. Allstot, D. Gangopadhyay, and D. J. Allstot, "Compressed sensing system considerations for ecg and emg wireless biosensors," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 6, no. 2, pp. 156–166, 2012.

[41] Z. Charbiwala, V. Karkare, S. Gibson, D. Markovic, and M. B. Srivastava, "Compressive sensing of neural action potentials using a learned union of supports," *Body Sensor Networks (BSN)*, pp. 53–58, 2011.

[42] Z. Zhang, T.-P. Jung, S. Makeig, and B. D. Rao, "Compressed sensing for energy-efficient wireless telemonitoring of noninvasive fetal ecg via block sparse bayesian learning," *IEEE Transactions on Biomedical Engineering*, vol. 60, no. 2, pp. 300–309, 2013.

[43] J. Zhang, Y. Suo, S. Mitra, S. P. Chin, S. Hsiao, R. F. Yazicioglu, T. D. Tran, and R. Etienne-Cummings, "An efficient and compact compressed sensing microsystem for implantable neural recordings," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 4, pp. 485–496, 2014.

[44] C. Bulach, U. Bihr, and M. Ortmanns, "Evaluation study of compressed sensing for neural spike recordings," *Engineering in Medicine and Biology Society (EMBC)*, pp. 3507–3510, 2012.

[45] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[46] P. H. Thakur, H. Lu, S. S. Hsiao, and K. O. Johnson, "Automated optimal detection and classification of neural action potentials in extra-cellular recordings," *Journal of Neuroscience Methods*, vol. 162, no. 1, pp. 364–376, 2007.

[47] R. Quiroga, Z. Nadasdy, and Y. Ben-Shaul, "Unsupervised spike detection and sorting with wavelets and superparamagnetic clustering," *Neural Computation*, vol. 16, no. 8, pp. 1661–1687, 2004.

[48] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.

[49] D. Gangopadhyay, E. G. Allstot, A. M. Dixon, K. Natarajan, S. Gupta, and D. J. Allstot, "Compressed sensing analog front-end for bio-sensor applications," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 2, pp. 426–438, 2014.

[50] J. Jia and K. Rohe, "Preconditioning to comply with the irrepresentable condition," *arXiv preprint arXiv:1208.5584*, 2012.

[51] S. A. Van De Geer, P. Bühlmann *et al.*, "On the conditions used to prove oracle results for the lasso," *Electronic Journal of Statistics*, vol. 3, pp. 1360–1392, 2009.

[52] A. Calabrese and L. Paninski, "Kalman filter mixture model for spike sorting of non-stationary data," *Journal of Neuroscience Methods*, vol. 196, no. 1, pp. 159–169, 2011.

[53] C. Pedreira, J. Martinez, M. J. Ison, and R. Q. Quiroga, "How many neurons can we see with current spike sorting algorithms?" *Journal of Neuroscience Methods*, vol. 211, no. 1, pp. 58–65, 2012.

[54] A. Levey and M. Lindenbaum, "Sequential karhunen-loeve basis extraction and its application to images," *IEEE Transactions on Image Processing*, vol. 9, no. 8, pp. 1371–1374, 2000.

[55] Y. Kim, W. Guo, B. V. Gowreesunker, N. Sun, and A. H. Tewfik, "Multi-channel sparse data conversion with a single analog-to-digital converter," *IEEE Journal*

on Emerging and Selected Topics in Circuits and Systems, vol. 2, no. 3, pp. 470–481, 2012.

[56] M. Hosseini Kamal, M. Shoaran, Y. Leblebici, A. Schmid, and P. Vandergheynst, "Compressive multichannel cortical signal recording," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4305–4309, 2013.

[57] M. Shoaran, M. M. Lopez, V. S. R. Pasupureddi, Y. Leblebici, and A. Schmid, "A low-power area-efficient compressive sensing approach for multi-channel neural recording," *International Symposium on Circuits and Systems (ISCAS)*, pp. 2191–2194, 2013.

[58] M. Shoaran, M. H. Kamal, C. Pollo, P. Vandergheynst, and A. Schmid, "Compact low-power cortical recording architecture for compressive multichannel data acquisition," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 8, no. 6, pp. 857–870, 2014.

[59] H. Ishwaran and J. S. Rao, "Spike and slab variable selection: frequentist and bayesian strategies," *Annals of Statistics*, vol. 33, no. 2, pp. 730–773, 2005.

[60] Y. Suo, M. Dao, T. Tran, U. Srinivas, and V. Monga, "Hierarchical sparse modeling using spike and slab priors," *International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3103–3107, 2013.

[61] H. Zou and T. Hastie, "Regularization and variable selection via the elastic

net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

[62] T. P. Minka, "Expectation propagation for approximate bayesian inference," *Uncertainty in Artificial Intelligence (UAI)*, pp. 362–369, 2001.

[63] D. A. Henze, Z. Borhegyi, J. Csicsvari, A. Mamiya, K. D. Harris, and G. Buzsáki, "Intracellular features predicted by extracellular recordings in the hippocampus in vivo," *Journal of Neurophysiology*, vol. 84, no. 1, pp. 390–400, 2000.

[64] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 11, pp. 2765–2781, 2013.

[65] K. Engan, S. O. Aase, and J. H. Husøy, "Multi-frame compression: Theory and design," *Signal Processing*, vol. 80, no. 10, pp. 2121–2140, 2000.

[66] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, 2003.

[67] S. Shekhar, V. M. Patel, H. V. Nguyen, and R. Chellappa, "Generalized domain-adaptive dictionaries," *Computer Vision and Pattern Recognition (CVPR)*, pp. 361–368, 2013.

[68] Q. Qiu and G. Sapiro, "Learning transformations for clustering and classification," *arXiv preprint arXiv:1309.2074*, 2013.

[69] J. Mairal, G. Sapiro, and M. Elad, "Learning multiscale sparse representations for image and video restoration," DTIC, Tech. Rep., 2007.

[70] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. Le-Cun, "Learning invariant features through topographic filter maps," *Computer Vision and Pattern Recognition (CVPR)*, pp. 1605–1612, 2009.

[71] K. Skretting and J. H. Husøy, "Texture classification using sparse frame-based representations," *EURASIP Journal on Applied Signal Processing*, vol. 2006, no. 1, pp. 102–102, 2006.

[72] K. Engan, K. Skretting, J. Herredsvela, and T. O. Gulsrud, "Frame texture classification method (ftcm) applied on mammograms for detection of abnormalities," *International Journal of Signal Processing (IJSP)*, vol. 4, no. 2, pp. 593–603, 2007.

[73] F. Rodriguez and G. Sapiro, "Sparse representations for image classification: Learning discriminative and reconstructive non-parametric dictionaries," DTIC, Tech. Rep., 2008.

[74] G. Zhang, Z. Jiang, and L. S. Davis, "Online semi-supervised discriminative

dictionary learning for sparse representation," *Asian Conference of Computer Vision (ACCV)*, pp. 259–273, 2013.

[75] Y. Zhang, Z. Jiang, and L. S. Davis, "Learning structured low-rank representations for image classification," *Computer Vision and Pattern Recognition (CVPR)*, pp. 676–683, 2013.

[76] D. Barchiesi and M. D. Plumbley, "Learning incoherent subspaces: Classification via incoherent dictionary learning," *Journal of Signal Processing Systems*, vol. 79, no. 2, pp. 189–199, 2014.

[77] C. Yu-Tseh, A. Mohsen, R. Ajit, and H. Jeffrey, "Block and group regularized sparse modeling for dictionary learning," *Computer Vision and Pattern Recognition (CVPR)*, pp. 377–382, 2013.

[78] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," *Computer Vision and Pattern Recognition (CVPR)*, pp. 3360–3367, 2010.

[79] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Structured sparsity through convex optimization," *Statistical Science*, vol. 27, no. 4, pp. 450–468, 2012.

[80] C. Studer and R. G. Baraniuk, "Dictionary learning from sparsely corrupted or compressed signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 3341–3344.

[81] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[82] J. Yang and Y. Zhang, "Alternating direction algorithms for l1-problems in compressive sensing," *SIAM Journal on Scientific Computing*, vol. 33, no. 1, pp. 250–278, 2011.

[83] Y. Nesterov, "Gradient methods for minimizing composite objective function," Center for Operations Research and Econometrics (CORE), Catholic University of Louvain, Tech. Rep., 2007.

[84] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[85] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach, "Proximal methods for hierarchical sparse coding," *The Journal of Machine Learning Research*, vol. 12, pp. 2297–2334, 2011.

[86] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski, "Optimization with sparsity-inducing penalties," *arXiv preprint arXiv:1108.0775*, 2011.

[87] D. A. Spielman, H. Wang, and J. Wright, "Exact recovery of sparsely-used dictionaries," *arXiv preprint arXiv:1206.5882*, 2012.

[88] R. Jenatton, R. Gribonval, and F. Bach, "Local stability and robustness of sparse dictionary learning in the presence of noise," *arXiv preprint arXiv:1210.0685*, 2012.

[89] K. Schnass, "On the identifiability of overcomplete dictionaries via the minimisation principle underlying k-svd," *arXiv preprint arXiv:1301.3375*, 2013.

[90] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 6, pp. 643–660, 2001.

[91] A. M. Martinez, "The ar face database," *CVC Technical Report*, vol. 24, 1998.

[92] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.

[93] Y. C. Eldar, P. Kuppinger, and H. Bolcskei, "Block-sparse signals: Uncertainty relations and efficient recovery," *IEEE Transactions on Signal Processing*, vol. 58, no. 6, pp. 3042–3054, 2010.

[94] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani, "A sparse-group lasso,"

Journal of Computational and Graphical Statistics, vol. 22, no. 2, pp. 231–245, 2013.

[95] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Transactions on Information Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.

[96] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," *Computer Vision and Pattern Recognition (CVPR)*, pp. 2169–2178, 2006.

Vita

Yuanming Suo received the B.S. degree in Electronic Information Science and Technology from Sun Yat-sen University, Guangdong, China, and the M.S.E. degree in Electrical and Computer Engineering from the University of Alabama in Huntsville, AL, USA. Currently, he is working towards the Ph.D. degree in the Digital Signal Processing Lab at the Johns Hopkins University, Baltimore, MD, USA. His research focuses on developing compressed sensing and dictionary learning algorithms for applications in biomedical signal analysis and image classifications. His other research interest include image analysis in medical/remote sensing applications.